WORDPRESS 3.0 SISÄLLÖNHALLINTAJÄRJESTELMÄNÄ ASIAKASKÄYTTÖÖN

Case Road Dream

Satu Suomi

Opinnäytetyö
Lokakuu 2010
Tietojenkäsittely
Proakatemia
Tampereen ammattikorkeakoulu

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Proakatemia suuntautumisvaihtoehto
SUOMI, SATU: WordPress 3.0 sisällönhallintajärjestelmänä asiakaskäyttöön, Case Road Dream

Opinnäytetyö 50 s., liitteet 0 s.
Lokakuu 2010

Heinäkuussa 2010 Road Dream antoi toimeksiantona luoda WordPress-blogialustaa käyttämällä yrityksensä www-sivut. Road Dream -tiimi oli syksyllä lähdössä polkupyörillä Tampereelta kohti Ateenaa ja sivujen tarkoituksena oli dokumentoida tämä matka blogijärjestelmää hyväksi käyttäen. Sivut koostuisivat niin dynaamisesta sisällöstä eli blogista kuin myös staattisesta sisällöstä, eli listasta sponsoreita. Road Dreamille oli tärkeää, että sivut olisivat helposti päivitettävissä, sillä niihin lisättäisiin sisältöä matkan aikana.

Sivuja tehdessä ilmeni rajoituksia WordPress-alustalla. Se ei täyttänyt kaikkia sisällönhallintajärjestelmän kriteerejä, joita sivujen tekemiseen olisi vaadittu, jotta ne olisivat olleet käyttäjäystävälliset. Kuitenkin tätä alustaa haluttiin käyttää, joten sen toimintatapoja piti lähteä edelleen muokkaamaan. Onneksi WordPress oli juuri sinä kesänä päivittänyt alustaansa 2.7-versiosta 3.0:aan. Tässä uudessa käyttöön otetussa versiossa oli jo sisään rakennettuna sisällönhallintajärjestelmälle vaadittavia ominaisuuksia. Ne eivät kuitenkaan olleet oletuksena esillä tai päällä vaan ne oli koodaamalla otettava käyttöön ja niille oli luotava oma käyttöliittymänsä.

Sivut valmistuivat ja ne otettiin käyttöön syyskuun alussa. Tuotoksen kylkiäisenä syntyi funktio-tiedosto, joka sisälsi koodin pätkiä, joiden avulla nämä ominaisuudet otettiin käyttöön. Tätä funktio-tiedostoa käyttämällä pystyisi helposti nämä sisällönhallintajärjestelmälliset ominaisuudet ottamaan käyttöön kaikissa WordPress 3.0 -alustalla toimivissa sivuissa. Funktio-tiedosto ei olisi ollut kuitenkaan se ainoa tapa toteuttaa nämä muutokset, vaan ne olisi voinut lisäosienkin avulla luoda. Kuitenkaan se ei olisi ollut oppimisen ja näiden ominaisuuksien ymmärtämisen kannalta mielekästä.

Funktio-tiedosto on osin hyvin Road Dream -keskeinen eikä kaikille sen ominaisuuksille luotu minkäänlaista käyttöliittymää. Jotta tätä tiedostoa voisi käyttää ihan missä tahansa blogissa muokkaamatta lähdekoodia, pitäisi sitä vielä tästä eteenpäin kehittää. Toisaalta se toimii hyvänä pohjana niille, jotka osaavat ja tuntevat PHP-koodia sekä WordPressin funktioita ja toimintatapoja.

Työssäni käsittelen WordPress 3.0:aa sisällönhallintajärjestelmänä kriittisestä näkökulmasta ja käyn läpi mitä se vaatii, että siitä saa kyseisen järjestelmän. Järjestelmän kehittämisen tärkeimpänä näkökulmana tässä työssä pidetään asiakaslähtöisyyttä ja helppokäyttöisyyttä.

Asiasanat: WordPress, sisällönhallintajärjestelmät, helppokäyttöisyys

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Option of Proacademy
SUOMI, SATU: WordPress 3.0 as Content Management System for the Clients, Case Road Dream

Bachelor's thesis 50 pages, appendices 0 pages
October 2010

In the July 2010 Road Dream gave an assignment to create a website using the WordPress blog platform. Road Dream was a six-man project to bicycle from Finland to Greece and to document the trip. They needed the blog for the documentation. The website was to consist of the blog and a few other static pages, including a list of sponsors. It also needed to be easy to use and update by the client without a need of an expert.

While doing this assignment it became obvious that the platform was not flexible enough for their content management. It lacked certain elements and properties. WordPress needed to be further manipulated to fulfil the criteria of content management systems. Fortunately, WordPress had released a new upgraded version which had these elements and properties build in the core. Still there needed to be more work done so that those properties could be used. They were not usable by default so some modifications needed to be done in the Functions file. The Functions file is where a developer can make modifications and enable certain existing properties without touching the actual core.

After studying more about WordPress and trying other real content management systems, it became clear what changes had to be done. While enabling these changes, it was important to maintain the usability of the system and make it light to use for the client.

The created Functions file can be used in any WordPress 3.0 blog. Although it has the required properties enabled, in some places they lack a proper user interface. So for this file to be usable for any user, it needs to be further modified. The user should need not to touch the code when it comes to content management systems.

Key words: WordPress, content management system, usability.

SISÄLLYS

1 JOHDANTO

Tämän työn aiheen ja idean sain syksyllä 2010 Road Dreamille tehdyn projektin aikana. Sain heiltä toimeksiantona WordPress-blogialustaa käyttämällä kehittää heidän omalle projektille www-sivut. Sivujen tarkoituksena oli toimia väylänä, jolla julkaista reaaliaikaista tietoa heidän projektinsa etenemisestä. Road Dream tiimin, joka koostui neljästä pyöräilijästä, sekä huoltojoukkoina toimineista auton kuljettajasta ja valokuvaajasta, tavoitteena oli pyöräillä Suomesta Kreikkaan ja dokumentoida matkan kulku. Road Dreamille oli tärkeää, että sivut olisivat helposti päivitettävissä, sillä niihin lisättäisiin niin dynaamista kuin staattista sisältöä matkan aikana.

Työstäessäni sivuja huomasin WordPress alustassa rajoitteita, joiden vuoksi helppokäyttöisyys saattaisi vaarantua ja se oli kuitenkin yksi Road Dreamin minulle asettamista tärkeimmistä kriteereistä. Tämän opinnäytetyön, myöhemmin pelkästään työ, aihe syntyi myös uteliaisuudesta alustaa kohtaan. Tätä uteliaisuutta lisäsi myös se, että WordPressistä oli tullut uusi versio 3.0, jonka luvattiin olevan lähempänä sisällönhallintajärjestelmää kuin blogialustaa. Näiden kaikkien yhteistuloksena aloinkin tutkimaan alustaa paremmin ja ryhdyin kehittämään sen ominaisuuksia eteenpäin. Tämä kehittäminen suurimmaksi osaksi tarkoitti näiden ns. uinuvien ominaisuuksien herättämistä. Ne oli siis koodaamalla otettava käyttöön, niille oli annettava ominaisuuksia ja niille oli myös luotava jonkinlainen käyttöliittymä.

Kaiken tämän tuloksena oli yksi funktio.php-tiedosto, joka sisältää teeman funktiot sekä oivan listan lisäosia, joiden avulla Road Dreamin sivut toimivat. Teeman funktioissa on paljon sellaisia asioita, jotka voi ottaa käyttöön kolmansien osapuolten lisäosilla, mutta ne ei aina taivu kaikkeen siihen mitä käyttäjä haluaisi. Pelkästään lisäosia käyttämällä en olisi myöskään oppinut ymmärtämään miten ne toimivat.

Kaikki käsitteet ja asiat, joita käsittelen luvuissa 1-4 ovat niitä, joiden kanssa työskentelin Road Dreamin sivuja tehdessäni. Käsittelen vain siis niitä asioita, jotka ovat käytännön toteutuksen kannalta oleellisia.

Luvussa 3. näin tarpeelliseksi verrata WordPressiä hieman muihin vastaaviin järjestelmiin. Vertailun avulla on helpompi luoda laajempi käsitys omasta aiheesta ja työstämästään alustasta. Vaikka esittelenkin jo tässä luvussa WordPressin, käsittelen sitä vasta luvussa 4 paremmin ja etenkin tätä 3.0 versiota. 5. luku, Case Road Dream kokoaa nämä kaikki käsitellyt asiat siten, että niiden toiminta esitellään reaalimaailmassa. Koska tätä projektia tehdessä kylkiäisenä syntyi toimiva teeman funktiot sisältävä tiedosto, on myös järkeä miettiä, miten sitä voisi jatkossa kehittää. Tätä aihetta käsittelen luvussa 6.

Lähteet ovat tässä työssä hyvin www-painoitteisia johtuen suurimmaksi osin siitä, että tämä aihe ja ala muuttuminen on hyvin hektistä. Kirjatietous on usein jo vanhentunutta. Poikkeuksena sääntöön käytin sisällönhallintajärjestelmiä käsiteltäessä Boikon Content Management Bible -kirjaa, jonka ensimmäinen painos ilmestyi jo vuonna 2001. Teknologiat muuttuvat ja päivittyvät, mutta niiden perusolemus ei juurikaan. Tämän vuoksi en nähnyt ongelmana käyttää niin massiivista, joskin vanhaa kirjaa lähteenä.

2 PERUSKÄSITTEITÄ

Tässä luvussa avataan työhön, WordPressiin ja www-suunnitteluun liittyviä peruskäsitteitä siten kuin niitä tuli Road Dreamin projektissa käsiteltyä.

2.1 HTML/XHTML

HTML (*Hypertext Markup Language)* ja XHTML (*eXtensible Hypertext Markup Language*) ovat www-ympäristössä käytettävän sisällön kuvaamiseen tarkoitettuja kieliä. Näiden avulla pystytään merkitsemään sisältöä ja määrittelemään tekstin eri osia. HTML:n avulla erotellaan toisistaan mm. otsikot, kuvat, linkit ja leipätekstit.

90-luvun alussa julkaistu HTML-kieli on koko ajan ollut alituisen kehityksen ja työstämisen alla. Tällä hetkellä käytössä on HTML 4 ja kehitystyön alla on HTML 5. Jokainen versio on määrittänyt ja edelleen tarkentanut standardeja, joiden mukaan sivuja tulisi luoda tai joiden mukaan selainten on sivut näytettävä.

XHTML on HTML-kielestä kehitetty merkintäsäännöiltään tiukempi variaatio, jonka päällimmäisenä tarkoituksena oli korvata aiempi versio. Tavoitteena oli luoda merkintäkieli, jota voisi käyttää muuallakin kuin tietokonepohjaisissa medioissa. XHTML toteuttaa XML-kielen sääntöjä. Sen kielioppi on tiukempaa ja tarkempaa; esimerkiksi tägit, jotka määrittävät mitä sisältö on, on kirjoitettava pienellä ja jokaisella aloittavalla tägillä on oltava vastaava lopetustägi. XHTML:stä on tullut 1.0 version lisäksi 2.0 (W3C 2010a).

WordPressin runko on kirjoitettu XHTML 1.0 -kielellä, jota se myös suosittelee käytettäväksi alustalle kehitettävien teemojen ja lisäosien kuvantamiseen. XHTML:n avulla on WordPressille ja sen teemoille luotu rakenne, jonka sisällä pyöritetään PHP:tä, jolla WordPressin toiminnot on luotu.

2.2 CSS

CSS (*Cascading Style Sheets*) on tyylittelykieli, jolla voidaan muotoilla HTML- tai XML-pohjaista sisältöä. CSS:n perimmäisin tarkoitus on ollut erotella sisältö ja tyylit toisistaan. Puhtaimmillaan tämä tarkoittaa sitä, että kuvauskielellä merkitty

sisältö ei sisällä minkäänlaisia tyylittelyjä vaan tiedostossa on viittaus ulkoiseen tyylitiedostoon, jossa tyylit sijaitsevat. Näitä tyylejä ovat mm. elementtien asettelu ja tekstin muotoilu (W3C 2010b).

2.3 PHP

PHP on avoimeen lähdekoodiin perustuva ohjelmointikieli, jota käytetään etenkin kehitettäessä dynaamisia www-sivuja. PHP-koodia voidaan lisätä HTML:n sisälle käyttäen <?php- aloitusmerkkiä ja ?>- lopetusmerkkiä, jotka erottelevat PHP-koodin muusta sisällöstä. Tämän ohjelmointikielen erityisenä ominaisuutena on se, että komennot suoritetaan palvelimella. Palvelin kääntää komennon html-kielelle, jonka selain sitten tulkitsee, kuten kuvasta 1 voidaan nähdä. Näin käyttäjä, joka selaa PHP-kieltä sisältäviä sivuja, ei pysty lähdekoodista näkemään, mitä komentoja on käytetty tiedon aikaansaamiseksi.

KUVA 1. PHP-sivujen muodostuminen käyttäjälle

PHP:n avulla on myös mahdollista koota useasta tiedostosta yksi tiedosto. Kun normaalisti HTML-sivu koostetaan kokonaisuudessaan yhteen tiedostoon, voidaan PHP:llä kutsua eri tiedostoja, joiden sisällön avulla voidaan koostaa yksi sivu. Esimerkiksi index.php-tiedosto voi koostua kolmesta eri tiedostosta; otsake.php, sisalto.php ja alapalkki.php. Index.php -tiedoston ei täten tarvitse sisältää kuin vain tietyt komennot, joiden avulla palvelin osaa näennäisesti liittää muiden tiedostojen sisällön yhdeksi HTML-tiedostoksi. Tämä helpottaa sivujen ylläpitoa siten, ettei jokaisen sivun tiedostoon tarvitse erikseen tehdä muutoksia, vaan ne voi hoitaa

keskitetysti yhteen tiedostoon. Koska selain ei käsittele PHP-kieltä, vaan palvelin hoitaa sen, niin tämän vuoksi kieltä käytettäessä on palvelimen tuettava PHP:tä (PHP 2010a).

PHP:n vahvuutena on sen kyky käsitellä tietokantoja. Kielen avulla onkin helppo luoda esimerkiksi nettikauppoja tuote- ja asiakasrekistereineen. Kielen avulla voidaan myös luoda erilaisia selainpohjaisia sovelluksia (PHP 2010b) .

WordPress on rakennettu PHP-kielellä, jota on osattava, jos tälle alustalle aikoo kehittää ja luoda teemoja tai lisäosia. WordPressin ytimeen on luotuna useita eri funktioita PHP-kielellä, jotka helpottavat kehitystyötä. Tavallisen käyttäjän, joka käyttää valmiita teemoja ja lisäosia, ei tarvitse osata PHP:tä, HTML:llää tai mitään muutakaan kieltä. Toimiakseen WordPress vaatii palvelimelta vähintään PHP 4.3:a (WordPress 2010a).

2.4 MySQL

MySQL on itsessään tietokanta ja sen avulla myös hallinoidaan SQl-relaatiotietokantoja . Tämän avulla on mahdollista luoda hakuja tietokannoista PHP:n tai Pythonin kaltaisia ohjelmointikieliä käyttäen. Järjestelmä myös pohjautuu vapaaseen lähdekoodiin, joten sen käyttö on ilmaista. MySQL onkin suosittu tapa hallinnoida erilaisten www-pohjaisten sisällönhallintajärjestelmien informaatiota ja esimerkiksi WordPress käyttää sitä tietojen varastoimiseen ja hakemiseen tietokannoista (MySQL 2010).

2.5 CMS

CMS (*content management system)* eli sisällönhallintajärjestelmä on yhteinen nimitys kaikille ohjelmille, joiden avulla pystytään hallitsemaan suuria määriä informaatiota. Tämä tieto voi sisältää niin tekstiä, kuvia, multimediaa kuin käyttäjätietoja ja yleensä ne on tallennettuna tietokantoihin. Tärkeää järjestelmille on käyttäjän kyky helposti hallinnoida tätä tieto; kerätä, luoda, muokata, julkaista tai poistaa sitä. (Boiko 2005, 71-73)

Näillä järjestelmillä on aina yksi tai useampi pääkäyttäjä, joiden vallassa on tämän tiedon hallinta sekä myös sen hallinta, ketkä tätä tietoa pääsevät näkemään tai muokkaamaan. Eli käyttäjähallinta on yksi sisällönhallinnan tärkeimmistä elementeistä (DocForge 2010).

Sisällönhallintajärjestelmät voidaan karkeasti jakaa kahteen kategoriaan. Julkisiin www-pohjaisiin järjestelmiin eli WCM:iin(*Web Content Management*) ja organisaatioiden sisäisiin selaimesta riippumattomiin järjestelmiin eli ECM:iin (E*nterprise Content Manafement).* (Boiko 2005, 79)

2.6 WCM-järjestelmät

Www-pohjaiset sisällönhallintajärjestelmät usein käsitetään kattavan koko sisällänhallinnallinen maailma. Jo pelkästään tämä kategoria sisältää erilaisia järjestelmiä.

Nimellisten järjestelmien avulla kyetään itse ohjelmassa luomaan sivuille sisältöä niin, että samalla sisältöön tehtyjä muutoksia pystytään seuraamaan reaaliaikaisesti todellisessa ympäristössään. Tällaista sisällönmuokkaamista kutsutaan WYSIWYG:ksi. Termi tulee sanoista What You See Is What You Get (suom. sen minkä näet sen saat). Käyttäjä siis pystyy koko ajan reaaliaikaisesti näkemään muutokset todellisessa ympäristössään.

WCM-järjestelmien tarkoituksena on tehokkaasti kasvattaa tunnettuutta, mahdollistaa asiakirjojen ja informaation julkaiseminen verkon välityksellä sekä luoda väylä, jolla hallita tietoa laitteesta ja paikasta riippumatta ilman, että käyttäjän tarvitsee osata ja ymmärtää teknisiä asioita (CMS critic 2010).

2.7 ECM-järjestelmät

Yrityskäyttöön tarkoitettujen sisällönhallintajärjestelmien tarkoitus on tuottaa helppokäyttöinen järjestelmä, joka sisältää yritystoiminnalle tärkeät strategiat sekä työkalut. Näiden avulla voidaan hallita ja julkaista yritystoimintaan liittyvää sisältöä sekä yhdistää eri toiminnot yhden hallinnan alaisuuteen. Hallitseminen tässä tarkoittaa tietojen, kuten asiakasrekisterin ja asiakirjojen, kuten sopimusten

tallettamista, muokkaamista sekä säilyttämistä. ECM-järjestelmien avulla ei siis pelkästään hallinnoida informaatioita sen tallettamista varten vaan, nämä työkalut on tarkoitettu yritystoiminnan sekä työskentelyn tehostamiseksi vähentämällä työvaiheita, järkeistämällä toimintaa sekä keskittämällä toiminnat yhteen, tarvittaessa. Sillä aina ei ole tarpeen luoda koko yrityksen kattavaa tietohallintoa, jolloin toiminnallisuuksia voidaan ottaa käyttöön pelkästään yrityksen tarpeen mukaan. Järjestelmille ominaista on myös niiden huomaamaton käytettävyys ja intergoiduissa järjestelmissä saumaton liikkuminen osioista toiseen. (Boiko 2005, 81)

Yritysten sisällönhallinta elää usein sulassa sovussa www-pohjaisten sisällönhallintajärjestelmän kanssa. Nämä täydentävätkin toisiaan siten, että WCM-järjestelmän avulla yritys pystyy luomaan julkaisuja, ylläpitämään www-sivujaan sekä pääsee käsiksi yritystoiminnan tietoihin yrityksen ulkopuolelta. Kuitenkaan ECM-järjestelmiä ei pidä sekoittaa WCM-järjestelmiin vaikka nämä ovatkin toisiinsa intergoitavissa. Tavanomaisia paikkoja löyttä ECM-järjestelmiä yhdistettynä WCM-järjestelmiin ovat esimerkiksi koulut ja näiden extra- sekä intranetit, kirjastojärjestelmineen sekä opiskelijatietokantoineen. (Kampffmeyer 2006, 2)

3 WCM-JÄRJESTELMÄT

Tässä luvussa esitellään kolme erillaista www-sisällönhallintajärjestelmää: Drupal, Joomla ja WordPress.

3.1 Drupal

Drupal sai alkunsa 2000-luvun alussa. Opiskelijakaverukset Buytaert ja Snijder olivat jakaneet langattoman verkon keskenään ja heille oli kehittynyt tarve jakaa tietoa keskustelupalstan tyylisellä alustalla, jossa toisen ei tarvitsisi olla reaaliaikaisesti paikalla. Buytaert kehitti drop.org:n tätä varten. Kehitystyössään hän sai apua kanssaopiskelijoiltaan, jotka olivat liittyneet palstan käyttäjiksi. Opiskeluiden päätyttyä Buytaert, Snijder sekä muut opiskelijakaverit päättivät jatkaa yhteydenpitämistä sen avulla. Buytaert jatkokehitti droppia eteenpäin julkaisten alustaa pyörittäneen ohjelman vuonna 2001. Tuo ohjelma tunnetaan siis nykyisin Drupalina (Drupal, 2010).

Drupal on hyvin kevyt alusta. Se on selvästi Joomlaa ja WordPressiä kevyempi ja sen asentaminen palvelimelle on vaivatonta. Keveys näkyy myös käyttöliittymässä, joka tosin voi olla alkuunsa hämäävä. Käyttöliittymän ulkonäkö on sama kuin sivustossa itsessään, tosin tämän voi asetuksista muuttaa mieleisekseen. Drupaliin on myös löydettävissä pelkästään käyttöliittymälle tarkoitettuja tyylejä, joten jos haluaa pysytellä vanhassa tutussa, niin sillekin löytyy vaihtoehtoja. Käyttöliittymän muokkaus on hyvin tärkeä ominaisuus siinä vaiheessa, kun sivuja tehdään asiakkaalle, ja sivusto halutaan brändätä niin edestä kuin takaa.

Drupal muistuttaa monin tavoin Joomlaa. Se on avoimeen lähdekoodin perustuva sisällönhallintajärjestelmä, jossa on mahdollista luoda omia sisältöjä sekä lisäosina toimivat tutut moduulit, joiden paikat määritetään koodiin asetettujen ”osoite” komentojen avulla. Itseasiassa Drupalia pidetään enemmänkin puitteena sisällönhallintajärjestelmälle. Sillä voidaan nopeasti asennuksen jälkeen luoda www-sivut tai sitten sitä voidaan kehittää eteenpäin verkostosivuksi tai lähes miksi tahansa. Drupal tarjoaa puitteet luoda mitä mielikuvitus keksii.

Toisin kuin Joomla, Drupal tarjoaa monia ominaisuuksia kovakoodattuna alustaan. Tämän lisäksi sille löytyy myös useita moduuleita, joiden ansiosta käyttäjän ei tarvitse osata HTML- tai PHP-kieliä. Drupalin perusominaisuuksiin kuuluu mm. tarinasisältöä (story), jolla voidaan esittää esimerkiksi sivustoa koskevia uutisia tai tiedotteita, sivusisältöä (page), blogiartikkeleita ja foorumeita. Drupalilla pystytään tuottamaan mm. verkostosivuja, sanakirjoja kuin myös tavallisia blogeja tai laajoja yrityssivuja.

Drupalista näkee selkeästi, että se on kehitetty tarpeeseen ja sitä luodessa on käytetty käyttäjäkokemuksia hyväksi. Alustan vakavaraisuuden näkee selvästi siinä, kuinka hyvin se kestää radikaaleja muutoksia ja uusia ominaisuuksia joita jokaisesta päivityksestä löytyy.

3.2 Joomla

Joomla aloitti Mambona. Mambo on kaupallinen sisällönhallintajärjestelmä, ja sitä kehittämässä ollut projektiryhmä erkaantui siitä 2005-vuonna erimielisyyksien vuoksi. Tämä ryhmä lähti jatkokehittämään avoimen lähdekoodin periaatteella toimivaa Joomlaa Mambon pohjalta (Joomla, 2010).

Joomla on hyvin erilainen järjestelmä WordPressiin verrattuna. Kun WordPress luotiin ensisijaisesti blogeille, on Joomla puhdasverinen sisällönhallintajärjestelmä. Se käyttää osioita (sections), kategorioita ja artikkeleita sisällön organisoimiseen sekä moduuleita lisäominaisuuksien lisäämiseen.

Joomlan kantavin voima on moduuleissa. Kuten Drupalissa ja WordPressissä, tälle alustalle sivuja tehdessä määritetään koodiin Drupalin valmiiksi nimeämiä paikkoja/koukkuja, jotka toimivat osoitteina järjestelmälle. Näiden avulla pystytään käyttöliittymän kautta liittämään moduuleita eri paikkoihin ilman, että niiden toiminallisuudet pitäisi koodata suoraan koodiin.

Joomlaan on luotu monia eri moduuleita, joiden avulla pystytään luomaan monia eri lisäominaisuuksia. Ilman näitä alusta onkin hyvin riisuttu, sisältäen vain artikkelin luomiseen vaadittavat osat. Kuitenkin ne asiat, jotka ovat vasta nyt WordPressissä kohdallaan, ovat olleet kunnossa Joomlassa jo pidemmän aikaa.

Tällaisia ominaisuuksia ovat valikoiden luonti, jolla pystytään tekemään useita erilaisia navigaatioita sivuille, ja kustomoidun sisällön tuottaminen.

3.3 Wordpress

WordPress on sisällönhallintapalvelu, joka voidaan ottaa käyttöön joko asentamalla ohjelmisto omalle palvelimelle tai rekisteröitymällä WordPress.com:n käyttäjäksi ja luomalla sivut heidän palvelimella sijaitsevalla WordPress ohjelmistolla.

WordPress on luotu ensisijaisesti blogialustaksi. Sen huomaa mm. käyttäjärooleissa sekä siinä, että sisällön tyyppejä on oletusarvoisesti vain kaksi: artikkelit ja sivut. Sisällönhallintaa määrittää hyvin vahvasti se, että on mahdollista luoda erityyppistä sisältöä, kuten kuvakatalogi, tai että on mahdollista luoda suurempia kokonaisuuksia sisällölle kuten asiakkaat. Eli ei ole pelkästään sivuja ja artikkeleita, vaan voisi olla vaikka asiakkaat- ja tuotteet-sisältöä ja tuotteet-sisältö sisältäisi kuvallisen katalogin.

WordPress 3.0:n julkaisun jälkeen tämä on tullut askeleen lähemmäksi normaalia käyttäjää. Kustomoidun sisällön luominen vaatii yhä kuitenkin joko ulkopuolisen lisäosan käyttöä tai koodaamista.

WordPressin eittämättä suurin voimavara on sen yhteisössä. Codex-kirjasto, joka sisältää kirjoittajien tekemiä artikkeleita eri funktioista, koodin pätkistä ja niiden käyttötarkoituksista, on ollut minulle suureksi avuksi. Koska artikkelit on kirjoitettu käyttäjältä toiselle, ovat ne selkeitä ja helposti ymmärrettäviä sekä niistä löytää usein käytännöllisiä vinkkejä, jotka käyttäjät ovat itse kokeilemalla huomanneet hyödyllisiksi. Nämä ovat niitä asioita, jotka usein jäävät palvelun kehittäjältä huomaamatta tai tämä pitää niitä itsestäänselvyytenä. Yhteisönvoima myös näkyy Codex:n laajuutena. WordPressin kehittäjät eivät olisi koskaan pystyneet luomaan niin kattavaa manuaalia yksin (WordPress 2010b).

4 WORDPRESS 3.0, THELONIUS

WordPress 3.0 oli kesällä 2010 yksi odotetuimmista päivityksistä sen historiassa. Sen tuomien uudistusten myötä WordPressin ominaisuudet alkoivat muistuttamaan paljon enemmän sisällönhallinnan vaatimuksia. Thelonious-nimen saanut päivitys toi mukanaan kustomoidut sisältötyypit, taxonomian, valikot sekä tuen useamman WordPress-sivun verkostolle (WordPress, 2010c).

4.1 Historia

WordPress on vuonna 2001 Michel Valdrighi julkaiseman b2/cafeblogin johdannainen. b2/cafeblog, kuten WordPresskin, toimi PHP-kielellä ja varastoi tiedot MySQL-tietokantaan. Tämän blogialustan käyttäjiin kuuluivat myös Matt Mullenweg sekä Mike Little, WordPressin perustajat. Alustan vahvuutena pidettiin sen lähdekoodin avoimuutta verrattuna 2000-luvun alun muihin alustoihin (WordPress, 2010d).

Kuitenkin muutama vuosi b2:n julkaisun jälkeen Michael Valdrighi lopetti sen kehittämisen ja katosi maan alle. Matt Mullenweg kuitenkin näki alustassa mahdollisuuksia jatkokehitykselle. Hän kirjoittikin artikkelin blogiinsa, johon Mike Little vastasi ja ehdotti yhteistyötä uuden alustan kehittämisestä b2:n pohjalta. He eivät olleet ainoita, jotka olivat lähteneet kehittämään uutta järjestelmää vanhan pohjalta. Vuonna 2003 François Planque ryhtyi kehittämään b2evolutionia (Mixergy, 2009).

B2:ssa oli monia asioita jotka miellyttivät ja muutamia jotka eivät miellyttäneet Matt Mullenwegia. Nämä vähemmän miellyttävät asiat ovatkin niitä, jotka WordPressissä toimivat parhaiten. Alustan asentaminen on nopeaa, asetuksien muokkaaminen on vaivatonta ja teemojen kehittäminen helppoa (Mixergy, 2009).

Vapaa-ajallaan Matt kehitti WordPressiä. CNET, joka käytti WordPressiä omissa projekteissaan, palkkasikin hänet pian. Hän työskenteli CNETin palveluksessa hetken, kunnes päätti jatkaa omaa Automattic-projektiaan, jossa tarjosi ohjelmia yksityisille käyttäjille ilmaiseksi ja yrityksille maksua vastaan. Tässä projektissa työskentelee tällä hetkellä 50 ihmistä ympäri maailmaa, ja siitä on syntynyt mm.

WordPressin mukana tuleva Akismet-lisäosa, BB Press-foorumi ja WordPress.com-palvelu, jossa käyttäjä pystyi ilman omaa kotisivutilaa pitämään blogiaan. Foorumi ei ole saanut niin suurta suosiota kuin WordPress. Automattik käyttää kuitenkin sitä omien projektiensa kehittämiseen (Mixergy, 2009).

4.2 Toimintatapa

Tässä esitellään ne toiminnot, jotka määrittelevät hyvin vahvasti sen, miten WordPress käytännössä toimii.

4.2.1 Silmukka eli The Loop

WordPressin toiminta perustuu silmukkaan. Silmukka luo kyselyn artikkeleista ja esittää ne. Silmukoissa voidaan käyttää WordPressin template-tageja, joiden avulla saadaan artikkeleista irti lisätietoja, joita ovat mm. kirjoittaja, päivämäärä ja kategoria (Codex, 2010a).

Silmukan alussa tutkitaan, onko artikkeleita, ja jos on, niin kauan artikkeleita listataan. Silmukoiden sisältöä voidaan määritellä sekä sitä, kuinka monta artikkelia silmukka maksimissaan näyttää. Esimerkkikoodissa 1 on tarvittava koodi silmukan aloittamiseksi. Aluksi siis tutkitaan, onko artikkeleita, If(have_posts()), ja niin kauan kuin niitä on, while(have_posts()), tulostetaan ne, the_post().

```
<?php if ( have_posts() ) : while ( have_posts() ) : the_post();
?>
```

ESIMERKKIKOODI 1.

Tämän jälkeen määritellään, mitä tietoja artikkelista halutaan esitellä. Pelkkä the_post()-funktion käyttö ei siis riitä artikkeleiden näyttämiseksi, vaan se kutsuu artikkelista tietoja sen mukaan, mitä kehittäjä on itse koodiin määrittänyt. Artikkelista tarvitsee minimissään hakea artikkelin tekstisisältö the_content()-funktiolla, lyhyt ote the_excerpt()-funktiolla ja otsikko the_title()-funktiolla.

Silmukoita voi olla myös useampia kuin yksi. Blogin sivulla voi olla pääsilmukan, joka listaa artikkelit, lisäksi kategoria-silmukka, joka listaa artikkelit tietystä kategoriasta. (Codex, 2010b).

4.2.2 Actions & filters

Toiminnallisilla koukuilla (action) WordPress osaa käynnistää niihin määritellyt funktiot oikean tapahtuman aikana. Kehittäjä käyttää näitä joko lisäosissa tai teeman funktioissa. Action-koukku (esimerkkikoodi 2.) ei tarvitse sisälleen kuin tiedon, missä tapahtumassa funktio suoritetaan, ja mikä se funktio on (Codex, 2010b).

```
add_action ( 'edit_post', ‘tee_jotain’ );
```

ESIMERKKIKOODI 2.

Tämän jälkeen määritellään, mitä funktio sisältää. Tällä toiminnalla käyttäjän muokatessa artikkelia funktio tee_jotain suoritetaan.

Suodattimet (filters) ovat funktioita, jotka tietyllä hetkellä nappaavat tietokantoihin siirtyvän datan ja manipuloivat sitä säädetyllä tavalla. Esimerkkikoodin 3 suodattimeen voidaan luoda funktio, joka suodattaa kaikki rumat sanat kommenteista. Tämä funktio asetetaan suodattimen avulla suoriutumaan silloin, kun tietokantaan lisätään kommentti.

```
add_filter('comment_text','suodata_kirosanat');
```

ESIMERKKIKOODI 3.

Kuten toiminnallisessa koukussa, myös suodattimessa täytyy määritellä minimissään mihin toimintoon se on sidottu ja minkä funktion se käynnistää.

Toiminnallisten koukkujen sekä suodattimien avulla voidaan myös poistaa käytöstä WordPressin sisäisiä toimintoja. (Codex, 2010c).

4.2.3 Teeman funktiot

Functions.php- tiedostolla voidaan ottaa käyttöön ominaisuuksia WordPressissä. Se toimii eräänlaisena lisäosana, johon on kuitenkin mahdollista tallettaa useita eri funktioita. WordPress tarkastaa käynnistyessään tiedoston olemassa olon, ja jos se on olemassa, niin suorittaa tiedoston sisällön (Codex, 2010d).

4.3 Ominaisuudet

Ominaisuuksissa esitellään, mitä WordPressin sisältä löytyy, ja miten niitä voidaan käyttää.

4.3.1 Sisältö

WordPressissä on kahdenlaista sisältöä, sivuja ja artikkeleita. Sivut sisältävät tavallisten www-sivujen tapaan staattisen sisältöä ja artikkelit ovat dynaamista sisältöä blogissa. Toisin kuin sisällönhallintajärjestelmissä yleensä, WordPressissä ei ole oletuksena mahdollista luoda uusia sisältötyyppejä. Versio 3.0 toi kuitenkin kehittäjien käyttöön kustomoidut sisältötyypit, jotka pystyy ottaa käyttöön teeman funktioissa. Tällä tavoin voidaan vaikka luoda tuotekatalogi tuotesisältötyypillä. Tälle sisältötyypille voidaan antaa kustomoituja kenttiä, joiden tietoja voidaan käyttää tuotesivulla hyväksi (Metcalf 2010).

4.3.2 Kategoriat, avainsanat ja kustomoitu luokittelu

Artikkeleita luokitellaan WordPressissä kategorioilla. Jokaisella artikkelilla on oltava vähintään yksi kategoria ja kaikkien kategorioiden on oltava uniikkeja. Näiden avulla voidaan luoda sivuille käyttäjää tukevia valikoita tai sivuja. Jos käyttäjä valitsee artikkelissa olevan kategorian, avaa se sivun jossa listataan kaikki artikkelit joilla on se sama kategoria.

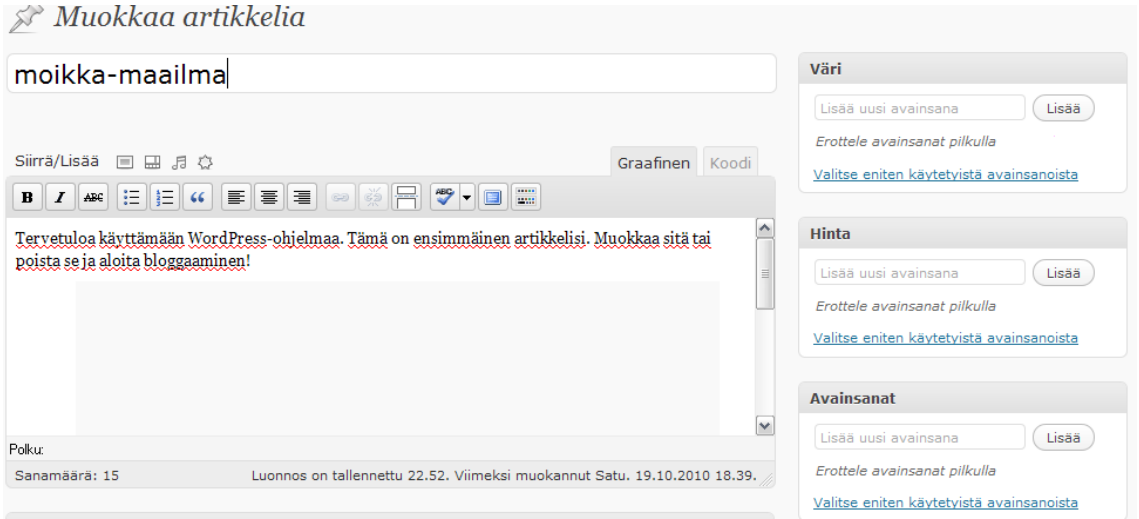
Yksi yleisimmistä tavoista käyttää kategorioita, on luoda arkisto artikkeleista kategorioittain. Kategorioille voidaan luoda hierarkioita, jotka mahdollistavat monimuotoisemman artikkelien luokittelun. Esimerkiksi käyttöjärjestelmä

kategoria voi sisältää alakategoriat iOS, Windows, Android, ja nämä voivat sisältää omia kategorioitaan.

Avainsanat eivät ole kategorioita. Nämä eri tavat luokitella artikkeleita saattavat hämmentää aloittavaa WordPressin käyttäjää. Avainsanat määrittelevät artikkelia enemmän kuin kategoriat ja tarjoavat lisä tietoa sekä ylimääräisen tavan käyttäjille liikkua sivuillasi ja etsiä tietoa (Codex, 2010e).

WordPress 3.0 mahdollisti kustomoitujen luokittelujen (custom taxonomies) luonnin. Avainsanoille ja kategorioille ei ole normaalisti omaa luokitusta. Eli artikkelilla ei voida antaa avainsanoja, joiden tyyppi on etukäteen määritelty. Esimerkiksi sivuille voidaan luoda tuotekatalogi, johon artikkelien avulla lisätään tuotteet. Tällöin tarvitaan myös tapoja luokitella tuotteita niin, että kategorioilla tai avainsanoilla on jo oma määritelty tyyppinsä. Toisin sanoen, artikkelien hinnoiksi ja väreiksi voidaan antaa omia avainsanoja (Kuva 1). Http://popcritics.com/-sivusto on erittäin hyvä esimerkki siitä, miten luokittelulla pystytään luomaan helposti käytettävä informatiivinen sivusto. Kustomoidut luokittelut eivät niinkään käy ilmi sivuilla kävijälle, vaan niillä on suurempi vaikutus WordPressin käytettävyyteen ja tiedon hallintaan.

Normaalisti käyttöliittymässä on vain yksi paikka kategorioille ja avainsanoille. Artikkelien lisäys sivulla olisi kategoria ja avainsana laatikoiden lisäksi omat paikat värille ja hinnalle (Kaiser 2010). Kuvassa 2 on esimerkki siitä miten kustomoidut luokittelut käytännössä näkyvät kirjoittajan puolella hallintapaneelissa.



KUVA 2. Kustomoidut luokittelut WordPress 3.0:ssa

Kustomoituja luokitteluja voidaan luoda joko rekisteröimällä tarvittavat luokat teeman funktioissa tai asentamalla sopiva lisäosa. Rekisteröintiä varten määritellään, millä nimellä luokittelua kutsutaan WordPressissä, eli miten se tallennetaan esimerkiksi tietokantoihin. Kehittäjän on tiedettävä, mitä sisältöä luokitellaan; artikkeleita, sivuja vai kustomoitua sisältöä (Esimerkkikoodi 4). Luokittelun ominaisuuksiksi voidaan antaa kyky hierarkisoida, voidaanko sitä hakea kyselyillä, ja jos käyttäjä valitsee luokittelun, onko sen aikaansaama osoite ystävällinen (Negoita 2010).

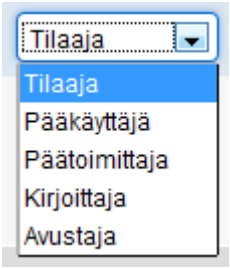
```
register_taxonomy(
      'color',
      'post',
      array(
            'hierarchical' => {true},
            'label' => 'Käyttäjälle näkyvä',
            'query_var' => {true},
            'rewrite' => {true }
      )
);
```

ESIMERKKIKOODI 4

4.3.3 Käyttäjät

WordPressin asennuksessa luodaan pääkäyttäjä, jolla on täydellinen valta alustan suhteen. Ennen versiota 3.0, tämän käyttäjän nimi oli ennalta valittu Admin. Tästä syystä käyttäjä joutui usein luomaan pääkäyttäjätilin lisäksi itselleen kirjoittajatilin, jonka nimi näkyisi artikkeleissa Adminin sijasta. Se että pääkäyttäjän tunnusta ei voinut itse päättää, aiheutti myös vakavia tietoturvaongelmia. WordPress 3.0:sta lähtien asennuksessa käyttäjä pystyy itse määrittämään luotavan pääkäyttäjän nimen. (Codex, 2010f).

Koska WordPress on blogi, ovat käyttäjäroolit hyvin kustannusmaailmaväritteisiä. Pääkäyttäjän lisäksi löytyy editoija-, kirjailija- ja tilaajaroolit (Kuva 3). Jokaisen roolin vaikutusmahdollisuudet vähenevät sitä mukaa mitä alemmaksi käyttäjäportaikossa mennään.



KUVA 3. WordPressin käyttäjäroolit

4.3.4 Ulkoasu

WordPressissä sivujen ulkoasua voidaan muokata ladattavilla teemoilla. Joissakin versiota 3.0 varten tehdyissä teemoissa on luotuna käyttöliittymään käyttäjää varten ulkoasun muokkaamiseen tarkoitettuja asetuksia. Näillä voidaan mm. määritellä teeman värejä, taustakuvia, fontteja ja toisinaan asettelua.

4.3.5 Lisäosat

Lisäosilla on mahdollista luoda ylimääräisiä toimintoja niin käyttäjän käyttöliittymään kuin myös sivustoon itseensä. Lisäosia WordPressiin voidaan asentaa joko siirtämällä lisäosan kansio palvelimelle plugin-kansioon tai sitten suoraan käyttöliittymässä lisäosien hallinnasta.

4.3.6 Valikot

Aikaisemmin käyttäjä ei ole juurikaan pystynyt vaikuttamaan siihen, miten valikot ja navigaatiot muodostuvat koskematta teeman tiedostoihin. Esimerkiksi kategorioita ei kyennyt ennen 3.0:aa käyttämään valikoissa ilman, että koski navigaation koodiin. Nyt valikoita voi luoda useita, ne voivat sisältää alivalikoita ja kategorioita, sekä jos teemaan on määritetty useampi kuin yksi paikka, voidaan valikolle sekin määrittää.

5 CASE ROADDREAM.ORG

Road Dream on neljän miehen projekti. Heidän tavoitteenaan oli pyöräillä Suomesta Kreikkaan ja dokumentoida matkan kulku blogiinsa. He lähtivät matkaan 1.9.2010 ja olivat perillä Kreikan Olympia-vuorella 14.10.2010.

Seuraavana käsittelen heille tehtyä WordPress-alustalla toimivaa blogia, sen kehitystä sekä sen toimintaa.

5.1 Toteutus

Road Dreamin www-sivut toteutettiin heinäkuun 2010 ja syyskuun 2010 välisenä aikana. Tähän kuului helppokäyttöinen käyttöliittymä, toimiva valokuva- sekä videopohjainen blogi, reaaliaikainen kartta ja staattiset sivut. Osaan toiminnallisuuksista käytettiin valmiita lisäosia ja osaan luotiin omat.

5.2 Teeman funktiot

Tiedostoon rakennettujen funktioiden avulla pystyin karsimaan ohjausnäkymästä sekä artikkeleiden muokkausikkunoista sellaisia asioita, joita asiakkaan ei tarvinnut käyttää ja tällöin myös nähdä. Tähän tiedostoon rakensin myös kustomoidut sisältötyypit (Custom Post Types), kustomoidun luokittelun (Custom Taxonomies) sekä otin käyttöön joukon muita ominaisuuksia.

5.2.1 Sponsorien kustomoitu sisältötyyppi ja kustomoidut kentät

Road Dreamin sivuille haluttiin helposti päivitettävissä oleva lista tukijoista. Nimet yksityishenkilöistä ja logot yhteistyökumppaneina olleista yrityksistä ja organisaatioista. Tämän olisi voinut toteuttaa siten, että olisin tavanomaiseen tapaan lisännyt sivun, tässä tapauksessa Kiitokset-sivun, jonne olisi tekstieditorin avulla pystynyt kirjoittamaan tukijoiden nimet sekä lisäämään yhteistyökumppaneiden logot. Tämän tavan ongelmana olisi ollut ongelmallinen tyylittely sekä tukijoiden määrän kasvaessa niiden siisti hallinta. Oli myös tarpeen kyetä erottelemaan toisistaan tukijat ja yhteistyökumppanit.

Teeman funktioihin lisäsin custom_spons_post()-nimisen funktion (Esimerkkikoodi 5), jonka avulla määrittelin niin toiminnallisuudet kuin sen ominaisuudet rekisteröiden uuden sisältötyypin nimeltä sponsors. Tälle tyypille luoduissa valikoissa asiakas kykeni lisäämään sponsoreita tai listaamaan kaikki olemassa olevat aivan samalla tapaa kuin artikkelien kanssa toimittaessa. Sponsorien listauksessa käyttäjä pystyi muokkaamaan ja poistamaan sponsoreita ja lisäyksessä lisäämään kuten normaalisti. Yksinkertaistaakseni sponsorien luomissivua, poistin tosin kaikki ne turhat artikkelin muokkaamiseen tarvitut ominaisuudet, joita ei sponsoreiden hallinnointia varten tarvittu. Otsikon, kustomoidut kentät sekä pienoiskuvan jätin editoriin. Esimerkkikoodissa 5suomensin toiminnot sekä määrittelin niiden nimet, määräsin sisällön julkiseksi, asetin valikon sijainnin tavallisten artikkeleiden alapuolelle käyttöliittymässä sekä määritin tarpeelliset kentät.

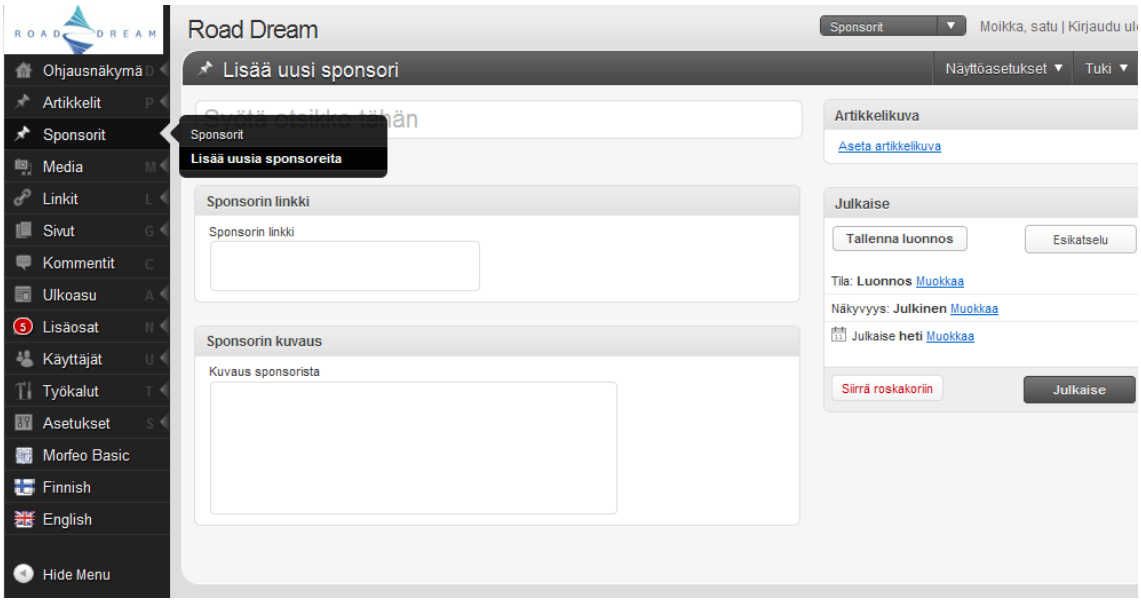
```
add_action('init', 'custom_spons_post');
function custom_spons_post(){
 $labels = array(
      'name' => 'Sponsorit',
      'singular_name' => 'Sponsori',
      'add_new' => 'Lisää uusia sponsoreita',
      'add_new_item' => 'Lisää uusi sponsori',
      'edit' => 'Muokkaa sponsoreita',
      'edit_item' => 'Muokkaa sponsoria',
      'new_item' => 'Uusi sponsori',
      'view' => 'Tarkastele sponsoria',
      'view_item' => 'Tarkastele sponsoria',
      'search_items' => 'Etsi sponsoria',
      'not_found' => 'Sponsori ei löytynyt',
      'not_found_in_trash' => 'Ei löytynyt roskiksesta'      );

 $args = array(
      'labels' => $labels,
      'description' => 'Sponsorit’,
      'show_ui' => true,
      'menu_position' => 5,
```

```
        'exclude_from_search' => false,
        'supports' => array('title','thumbnail',’custom_fields’) ,
        'public' => true,  );
register_post_type('sponsors',$args); }
```

ESIMERKKIKOODI 5.

Sponsorien lisäämissivulle luotiin kaksi kustomoitua kenttää (custom fields), Sponsorin linkki sekä Sponsorin kuvaus. Näiden lisäksi sivulta löytyy myös pakollinen otsikko-kenttä, johon laitetaan tukijan tai yhteistyökumppanin nimi ja artikkelikuva (tunnetaan myös pikkukuvana), johon yhteistyökumppanin kuva voidaan hakea.

Pakollisia kenttiä on vain yksi ja tämä on otsikko. Pelkän otsikon täyttäessä Kiitokset-sivulla näkyy vain tähän lisätty nimi. Jos otsikon lisäksi on valittuna kuva, ei sponsorista näy kuin tuo kuva. Ja jos linkki on lisätty, niin kuva tai nimi muuttuu hyperlinkiksi osoittamaan linkkikenttään lisättyyn osoitteeseen. Sponsorin kuvauskentän avulla oli alkuperäisen suunnitelman mukaan sisällettävä lyhyt tervehdys tukijalta. Tämä suunnitelma ei kuitenkaan toteutunut. Kuvassa 4 on näkymä ohjauspaneeliin ja uuden sponsorin lisäämissivun käyttöliittymään, jossa kustomoitujen kenttien toiminta näkyy käytännössä.



KUVA 4. Kustomoidut kentät WordPress 3.0:ssa

Kustomoitujen kenttien avulla pystyin luomaan asiakkaalle helpomman käyttöliittymän. Näiden avulla asiakkaan ei tarvinnut ihmetellä, mihin mikäkin tieto kuuluu. Nämä kentät rakensin jälleen teeman funktioihin lisäämien toiminnallisuuksien avulla.

Kustomoidut kentät vaativat kaksi erilaista toiminnallisuutta; Admin_init sekä save_post (Esimerkkikoodi 6). Näiden avulla pystytään kustomoituja kenttiä kutsumaan oikeassa paikassa. Admin_init-toiminto käynnistää funktion sponsor_box. Tämä funktio luo meta-tietoa sisältävät kentät: sponsor_desc ja sponsor_link, jotka taasen tallennetaan save_post-toiminnolla.

```
add_action("admin_init", "sponsor_box");
add_action('save_post', 'save_sponsor_desc');
add_action('save_post', 'save_sponsor_link');
```

ESIMERKKIKOODI 6. Metakenttien luontiin ja tallennukseen vaadittavat funktiot

WordPress käynnistää sponsor_box-funktion (Esimerkkikoodi 7) joka kerran, kun hallinnan puolella liikutaan. Tämä funktio ilmaisee, että add_meta_box()-funktioita suoritetaan kaksin kappalein, jos sivun sisältötyyppi on sponsors. Toinen meta-kenttä saa nimekseen Sponsorin kuvaus, jonka toiminnallisuus löytyy funktiosta sponsor_desc. Toinen saa nimekseen Sponsorin linkki ja sen toiminnat löytyvät funktiosta sponsor_link.

```
function sponsor_box(){
  add_meta_box("sponsor_desc-meta", "Sponsorin kuvaus", "spon-
sor_desc", "sponsors", "normal", "high");
  add_meta_box("sponsor_link-meta", "Sponsorin linkki", "spon-
sor_link", "sponsors", "side", "high");
}
```

ESIMERKKIKOODI 7. Metakenttien ominaisuuksien määrittely

Tässä luodaan määritykset funktiolle sponsor_desc. Ensin se liittää kyseisen sponsorin kuvauksen sen yksilölliseen ID:een. Tämän jälkeen se luo sponsor_desc-funktion (Esimerkkikoodi 8), eli sponsorin kuvaus -kentän tämän lisäämissivulle.

Jos tämän sponsorin kuvaukseen on lisätty tietoa, näytetään se. Jos ei, niin kuvaus on tyhjä. Molemmissa tapauksissa tätä kenttää voidaan muokata. Lopuksi sponsorin kuvaus –kenttä tallennetaan tietokantaan.

```
function sponsor_desc(){
  global $post;
  $custom = get_post_custom($post->ID);
  $sponsor_desc = $custom["sponsor_desc"][0];  ?>
 <p><label>Kuvaus sponsorista</label><br />
  <textarea cols="50" rows="5" name="sponsor_desc"><?php echo
$sponsor_desc; ?></textarea></p>
  <?php
}
function save_sponsor_desc(){
global $post;
$post_id = $post->ID;
```

ESIMERKKIKOODI 8.

Välttyäksemme inhottavalta autosave-bugilta, lisäämme toiminnon (esimerkkikoodi 9), joka päivittää meta-kentät siinä tapauksessa, että sponsorin muokkaussivu tallentaa itsensä automaattisesti. Ilman tätä eivät tiedot tallennu, sillä automaattinen tallennus ei ymmärrä tallentaa kustomoituja tietoja.

```
if ( defined('DOING_AUTOSAVE') && DOING_AUTOSAVE )
return $post_id;
update_post_meta($post->ID, "sponsor_desc",
$_POST["sponsor_desc"]);
}
```

ESIMERKKIKOODI 9. Autosave-bugin korjaaminen

Sponsorilinkin kentän luominen tapahtuu aivan samalla tapaa kuin sponsorin kuvauksen kentän luominen.

Kustomoitujen kenttien avulla pystyin helposti erottelemaan Kiitokset-sivulla tukijat ja yhteistyökumppanit toisistaan. Kiitokset–sivulle loin if-lauseen

(esimerkkikoodi 10), joka kyseli löytyykö artikkeleita, joiden sisältötyyppinä on sponsors. Olin aikasemmin säätänyt WordPressin asetuksista yhden sivun artikkelien kokonaismäärään ylärajaksi 8, ja jouduinkin määrittämään kyselyssäni, että niitä halutaan tällä kertaa näyttää maksimissaan 999 artikkelin verran. Luotin siihen, että tukijoita ei tule tätä enempää. Ensiksi loin kyselyn artikkeleista, ja tämän jälkeen kehitin kaksi silmukkaa, joissa tutkitaan artikkeleiden sisältöä.

```
<?php $sponsor_query = new WP_Query(array('post_type' => 'spon-
sors','posts_per_page' => 999)); ?>
```

ESIMERKKIKOODI 10.

Tätä kyseistä (Esimerkkikoodi 11) kyselyä ($sponsor_query) suoritetaan niin kauan kuin (while) artikkeleita löytyy tietokannasta. Tässä ensimmäisessä silmukassa partner-list-nimistä listaa (<ul id="partner_list”>) varten tarkistetaan, onko artikkelille määritelty pikkukuvaa, eli onko se tukija vai yhteistyökumppani. Jos kuvaa ei ole, jatketaan tässä silmukassa. Jos on, siirrytään seuraavaan.

```
<ul id="partner_list">
<?php while ($sponsor_query->have_posts()) : $sponsor_query-
>the_post(); if(!has_post_thumbnail()){?>
```

ESIMERKKIKOODI 11.

Jos kuvaa ei ole, on kyseessä tavallinen tukija. Tukijalla on mahdollista olla oma linkki ulkoiseen lähteeseen. Tässä vaiheessa tarkastetaan, onko tukijan sponsorilinkkikenttä täytettynä (Esimerkkikoodi 12). Jos on, se luo otsikon ympärille hyperlinkin, jonka osoitteeksi se asettaa sponsorilinkkikenttään määritetyn osoitteen.

```
<li>
 <?php $customField = get_post_meta($post->ID, 'sponsor_link',
true);
if(isset($customField["sponsor_link"])){?> <a href="<?php echo
get_post_meta($post->ID, 'sponsor_link', true); ?>"><?php
the_title();?></a>
```

ESIMERKKIKOODI 12.

Jos tukijalle ei ole määritettynä linkkiä, näytetään se sivuilla tavallisen tekstin tapaan (Esimerkkikoodi 13).

```
<?php }else{ the_title();}?>
 </li>
```

ESIMERKKIKOODI 13.

Kysely päätetään ja silmukka suljettiin (Esimerkkikoodi 14).

```
  <?php } endwhile; ?>
</ul>
```

ESIMERKKIKOODI 14.

Jos sponsorille on määritettynä pikkukuva, siirrytään suorittamaan tätä osaa koodista. Toisin kuin tukijoilla, kaikille yhteistyökumppaneilla tuli olla linkki omille sivuilleen. Nyt ei ollut tarpeen tarkistaa, onko linkkiä lisättynä sponsorilinkkikenttään. Yhteistyökumppanilista luotiin kutsumalla kuvaa ja muuntamalla sen hyperlinkiksi tämän sivuille (Esimerkkikoodi 15).

```
<ul id="sponsor_list">
  <?php if(has_post_thumbnail()) if ($sponsor_query-
>have_posts()) : while ($my_query->have_posts()) : $spon-
sor_query->the_post();
 if(has_post_thumbnail()){?>
<li>
        <a href="<?php echo get_post_meta($post->ID, 'spon-
sor_link', true); ?>">
        <?php the_post_thumbnail(featured-thumbnail);?></a>
   </li>
```

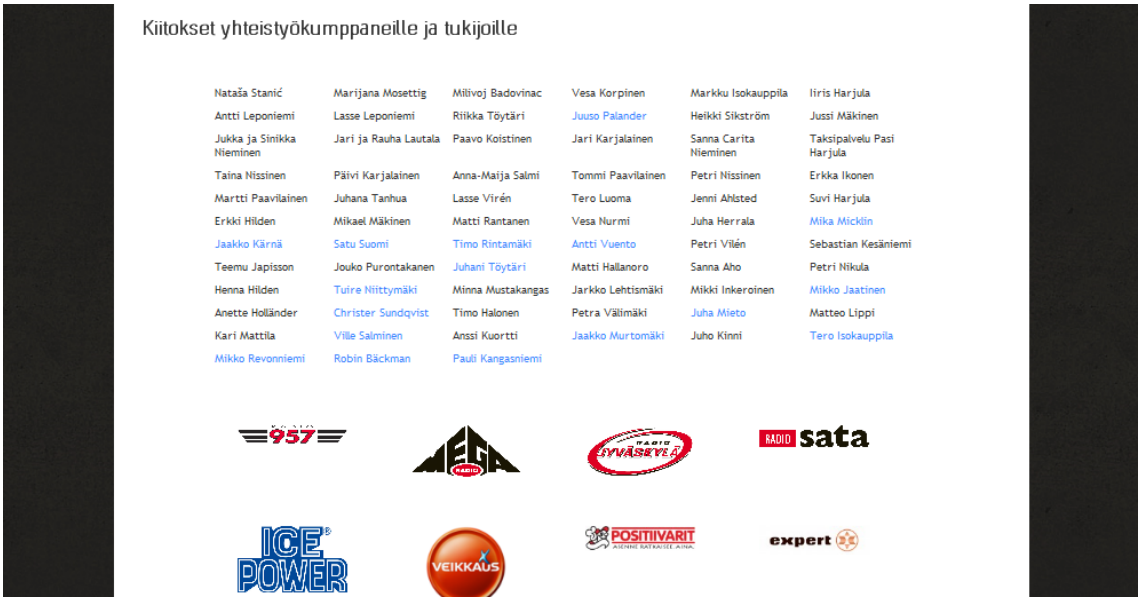
ESIMERKKIKOODI 15.

Kysely päätetään ja silmukka suljetaan (Esimerkkikoodi16).

```
<?php } endwhile; endif; ?>
```

ESIMERKKIKOODI 16.

Tällä tavoin kykenin käyttämään teeman funktiossa määrittelemääni sponsors-sisältötyyppiä käytännössä sekä manipuloimaan sitä ja sen avulla saatua tietoa. Näiden koodien lopputuloksena oli kaksi eriteltyä listaa, joissa toisessa oli tukijoiden nimiä ja toisessa yhteistyökumppanien logoja (Kuva 5).



KUVA 5. Kustomoitu sisältö toiminnassa Road Dreamin sivuilla

Asiakkaan käyttömukavuutta ajatellen oli myös järkevää manipuloida tapaa, jolla sponsorit listattiin. Kustomoituja palstoja varten tarvitsi luoda suodin sekä toiminto, jonka avulla jälleen WordPress osaisi oikeassa kohtaa suorittaa teeman funktioita. Suodattimella manage_edit-sponsors_columns (Esimerkkikoodi 17) käynnistettiin edit_sponsor_columns-funktio sivulla, joka listaa sponsorit (edit-sponsors). WordPressin ytimeen on määritetty muutama asia, joita voidaan näyttää näissä palstoissa. Näihin ei kuitenkaan kuulu thumbnail, eli pikkukuva ja links eli linkit. Näitä varten on sponsors_columns_display-funktiota käyttämällä määritettävä joukko toiminnallisuuksia. Tätä varten kutsutaan toimintoa

manage_posts_custom_column, joka ymmärtää näyttää sponsorien listaus sivulla nämä uudet palstat.

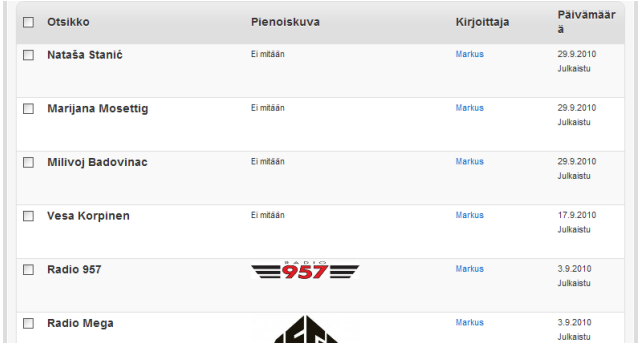
```
add_filter("manage_edit-sponsors_columns", "
edit_sponsor_columns");
add_action("manage_posts_custom_column",  "spon-
sors_columns_display", 10, 2);
```

ESIMERKKIKOODI 17.

Edit_sponsor_columns-funktiolla käsketään sponsorin listaussivun näyttää seuraavat palstat: cb, eli valintaruutu sitä varten, että voidaan massapoistaa tai -muokata sponsoreita, title otsikkoa varten, thumbnail-pikkukuvaa ja date-päivämäärää varten (Esimerkkikoodi 18). Näistä palstoja ei tarvinnut suomentaa, sillä nämä ovat oletus arvoja palstoille. Road Dreamin sivujen alustana käytetty suomennettu versio WordPress 3.0 sisältää suomennokset näihin oletusarvoisiin ominaisuuksiin (Kuva 6).

```
function edit_sponsor_columns($sponsors_columns){
$sponsors_columns = array(
      "cb" => "<input type=\"checkbox\" />",
      "title" => _x('Title', 'column name'),
      "thumbnail" => __('Thumbnail'),
      "date" => __('Date'));
}
```

ESIMERKKIKOODI 18.



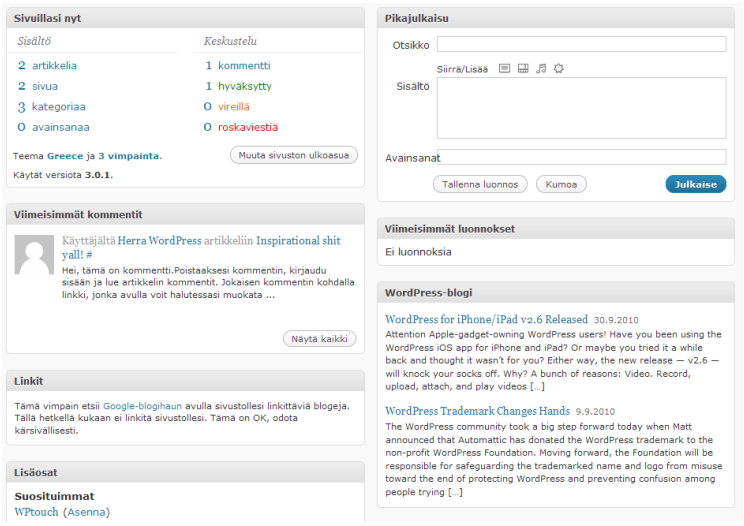
KUVA 6. Kustomoidun sisällön kustomoitu listaus

5.2.2 Käyttöliittymä, sen brändäys ja siistintä

Alusta asti asiakas toivoi siistiä ja helppokäyttöistä käyttöliittymää. Tämä sisälsi niin ohjauspaneelin valikoiden määrittämisen asiakkaalle sopivaksi kuin myös artikkeleiden, sponsorien sekä staattisten sivujen editorien ja ohjausnäkymän ominaisuuksien karsimisen. Valikoiden uudelleenjärjestelemisen sekä ominaisuuksien järkeistämisen lisäksi halusin mukaistaa käyttöliittymää Road Dreamin ilmeeseen sopivaksi. Tämä loisi asiakkaan kannalta yhteneväisen kokonaisuuden, joka näyttäisi täysin asiakkaalle räätälöidyltä. Tämä räätälöinti käsitti myös logojen lisäämisen niin käyttöliittymään kuin kirjautumisikkunaa.

Ihan ensimmäisenä ryhdyin tutkimaan lisäosia, joiden avulla saisin toteutettua ainakin osan näistä ominaisuuksista. Heti ensimmäisenä törmäsin Fluency-lisäosaan, joka uudelleen teemoitti koko käyttöliittymän. Tämä sopi hyvin väreiltään Road Dreamin ilmeeseen ja sen toiminnallisuuksiin kuului myös logojen lisääminen käyttöliittymään sekä kirjautumisikkunaan.

Uuden käyttöliittymän jälkeen ryhdyin tutkimaan miten ohjausnäkymän elementtejä pystyisi poistamaan. Ohjausnäkymä kun on se ensimmäinen sivu, joka aukeaa alustalle kirjauduttua ja se on oletuksena hyvin täynnä erillaisia elementtejä ja toiminnallisuuksia, joista osa on turhia ja hidastavat vain hallinnan lataantumista (Kuva 7). WordPress 3.0:ssa ei myöskään enää toimi se, että näitä menisi näyttöasetuksista manuaalisesti muokkaamaan. Ne eivät jostain syystä enää tallennu vaan ne pitäisi joka kerta tehdä uudestaan.



KUVA 7. WordPressin perinteinen ohjausnäkymän etusivu

Codexista selvisi, että remove_meta_box-funktiota voidaan käyttää turhien artikkeli yms. sivujen laatikoiden poistamiseksi. WordPressistä löytyy myös funktio unset($wp_meta_boxes['dashboard']['normal']['core']['poistettavan osan nimi']), jolla taasen ohjausnäkymän osasia voidaan karsia. Teeman funktioihin oli luotava lisää toiminnallisuuksia.

Lisäsin teeman funktioihin kaksi toimintoa, joista toinen kutsuu remove_dashboard_widgets-funktiota käyttäjän avatessa ohjausnäkymän, ja toinen joka käynnistää delete_meta_boxes-funktion käyttäjän liikkuessa sivuston hallinnan sivuilla. Ensimmäinen poistaa ohjausnäkymästä, jälkimmäinen poistaa turhat laatikot ja kentät artikkelin ja staattisten sivujen editoimis-sivuilta (Esimerkkikoodi 19).

```
add_action('wp_dashboard_setup', 'remove_dashboard_widgets' );
add_action('admin_init','delete_meta_boxes');
```

ESIMERKKIKOODI 19.

Delete_meta_boxes-funktion tarkoituksena on selvittää sivujen hallintaa selaavan käyttäjän käyttäjäryhmä ja sen mukaan sitten poistaa näkyvistä ominaisuuksia (Esimerkkikoodi 20).

```
function delete_meta_boxes() {
     //retrieve current user info
     global $current_user;
     get_currentuserinfo();
```

ESIMERKKIKOODI 20.

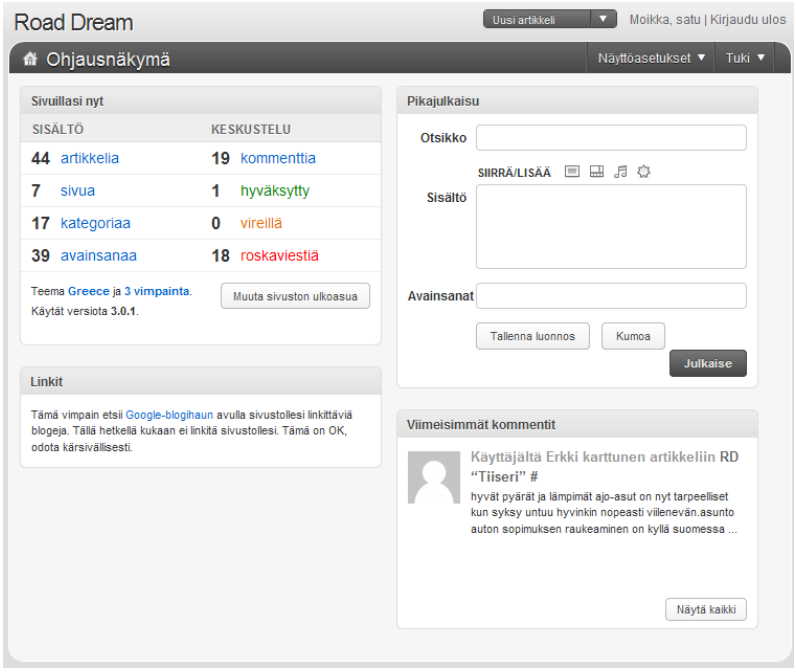
Käyttäjätason ollessa pienempi kuin 10, katoavat määritetyt laatikot ja kentät kaikkien käyttäjien näkyvistä, sillä taso 10:n on ylin pääkäyttäjätaso. Artikkelin muokkaamissivuilta poistettiin artikkelien kommentit-, viittaavat sivustot-, kirjoittajan tiedot- sekä ote artikkelista-laatikot. Sivujen muokkaussivuilta poistettiin kirjoittajan tiedot-, kommentit- sekä artikkelien maantieteellisen sijainnin määrittämiseen tarkoitettu Geo Mashupin kartta -laatikot (Esimerkkikoodi 21).

```
 if ($current_user->user_level < 10)
remove_meta_box('commentstatusdiv','post','normal');
…
remove_meta_box('geo_mashup_post_edit','page','normal');
}
```

ESIMERKKIKOODI 21.

Lisäosia asennettaessa saattaa aina ajoittain ohjausnäkymään asentua laatikoita ja toiminnallisuuksia, joita käyttäjä ei tarvitse. Yleensä Yoastin kehittämät lisäosat asentavat näitä, mutta myös oletuksena turhia tällaisia löytyy. Näiden poistamiseksi kutsutaan remove_dashboard_widgets-funktiota. Tätä varten on kuitenkin tiedettävä, mitä nimeä nämä turhakkeet käyttävät. Näiden nimet on löydettävissä WordPressin ytimestä wp-admin/includes-kansiosta tiedostosta nimeltä dashboard.php. Tuosta tiedostosta on myös mahdollista poistaa kaikki turhat, mutta järkevintä se on tehdä teeman funktioiden avulla. Näin mitkään WordPressin päivitykset eivät pysty vaikuttamaan tehtyihin muutoksiin.

Tällä kaikella saatiin aikaan kevyt etusivu, josta asiakas näki yhdellä vilkaisulla kaikki tarvitsemansa tiedot sivustaan (Kuva 8).



KUVA 8. Ohjausnäkymän etusivu tehtyjen muutoksen jälkeen

5.2.3 Sekalaiset ominaisuudet

Road Dreamin sivuille tehtyä teemaa varten oli teeman funktioihin lisättävä tuki muutamalle ominaisuudelle, jotka eivät ilman sitä muuten toimisi. Näiden avulla saatiin artikkelien editoimissivuille artikkelikuvat laatikko, jonka toimintaan kohokohdat osion pikkukuvat hyvin vahvasti tukeutuvat. Blogin artikkeleiden kuvien enimmäiskoon määritys onnistui myös teeman funktioissa. Näin eivät sivun leveyttä isommat kuvat riko ulkoasua. Myös sivuston navigaation käyttöliittymän käyttöön otto toteutetiin funktio-tiedoston avulla, kuten myös sivupalkkien rekisteröinti vimpainvalmiiksi.

Teemaan lisättiin monenlaisia kuvan kokoon liittyviä funktioita. Ensinnäkin mahdollistettiin pikkukuvien käyttö mm. lisäämällä add_theme_supportin-funktion avulla artikkelikuva-laatikko sivujen ja artikkelien muokkaussivulle. Tämän lisäksi luotiin enimmäiskoot kohokohtien ja sisällön kuville sekä määriteltiin niiden pitämään mittasuhteensa koosta huolimatta.

WordPress 3.0 mahdollisti sivujen navigoinnin ja valikoiden luomisen käyttöliittymän kautta. Tässä tarkastetaan, löytyykö reksiteröitävää valikkoa, ja jos löytyy, niin se rekisteröi sen. Tämän avulla ei tarvitse turhaan lisätä teeman tukea valikoille. Loin Road Dreamille yhden valikon, Päänavigaation. Index.php-tiedostossa kutsun tätä valikkoa wp_nav_menu()-funktiolla. Asiakas pystyy näin helposti muokkaamaan valikkoa käyttöliittymän avulla. Ja tällä tavalla myös pääkäyttäjä kykenee helposti muokkaaman ja luomaan lisää paikkoja joihin liittää valikoita.

Road Dreamin sivuilla on selkeä sivupalkki. Tässä sivupalkissa sijaitsevat linkit RSS-syötteisiin, sähköpostilistaan, radioraportteihin, uutiskirjeeseen sekä siihen myös listataan sekä facebookin, että twitterin tilapäivitykset. Nämä kaikki on määritettynä lisäosiin, jotka on lisätty sivupalkkiin WordPressin oman ulkoasu-sivun käyttöliittymän avulla. Tämä saadaan käyttöön, kun teeman funktioissa rekisteröidään ainakin yksi sivupalkki. Sivupalkille annettu Sivupalkki-nimi näkyy ulkoasu-sivulla ja id:n avulla tämä sivupalkki pystytään erottamaan muista sivupalkeista.

Jotta sivupalkkia voidaan käyttää, on sitä, kuten valikkoakin kutsuttava jossain kohtaa teemaa. Road Dreamin sivulla kutsun sivupalkkia WordPressin omalla get_sidebar()-funktiolla alapalkki-tiedostossa, jotta tärkein tieto, eli artikkelit lataantuisivat ensimmäisenä etusivulle. Tämä on tärkeää, koska Road Dreamin sivujen sisältö on raskasta ja kävijän kannalta on parasta priorisoida sisällön lataantuminen ennen muita.

5.3 Lisäosat

Sivustoa työstettäessä tein selvän päätöksen siitä, että en käyttäisi sivuilla lisäosia, jotka on tarkoitettu mahdollistamaan WordPress 3.0 uusien ominaisuuksien käyttöönoton. Näiden lisäosien avulla ominaisuudet kuten Custom Post Types ja valikot olisivat olleet käyttövalmiina silmänräpäyksessä. Halusin kuitenkin oppia ymmärtämään, miten nämä ominaisuudet rakentuvat tekemällä itse.

Itse koodasin pelkästään näitä uusia ominaisuuksia. Muut ominaisuudet otin käyttöön lisäosien avulla. Esittelen niin ne lisäosat, joita käytin sivuilla kuin myös ne jotka hylkäsin, ja joidenka tuomat ominaisuudet itse loin.

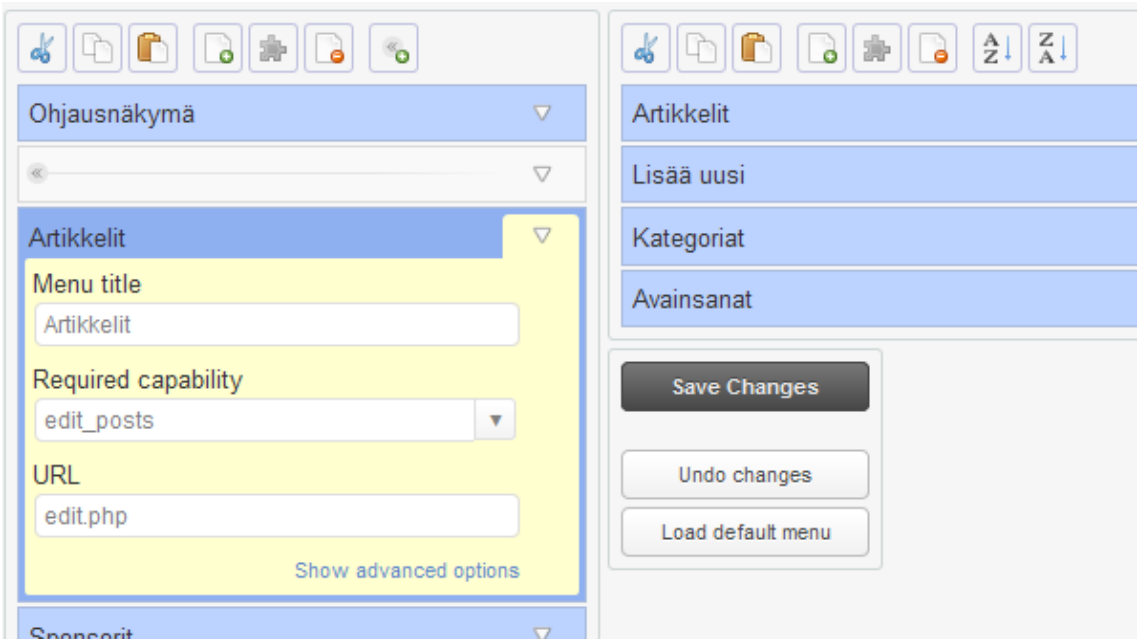
5.3.1 Admin Menu Editor

Admin Menu Editor (Kuva 9) on lisäosa, jonka avulla pystytään hallitsemaan käyttöliittymää ja sen ominaisuuksia käyttäjäryhmän mukaan. Sen avulla pystytään siirtämään ohjausnäkymän valikoita, piilottamaan niitä sekä muokkaamaan niiden näkyvyyttä käyttäjäryhmien mukaan (WordPress, 2010g).

Lisäosan otin käyttööni User Role Editoria kokeiltuani. Sen avulla olisin kyennyt hallitsemaan käyttäjiä ja käyttäjäryhmiä, mutta Admin Menu Editor osoittautui paremmaksi.

Road Dream sivuston sisällön tuottajat olivat toivoneet käyttöliittymää, joka olisi yksinkertainen, ja jossa ei olisi mitään heille turhaa tavaraa sotkemassa. Admin Menu Editorilla pystyinkin piilottamaan tiettyjä valikoita tuottajien näkyvistä. Pystyin myös tutkimaan, mitä sijainnillisia asetuksia näille lisäosille oli määritetty

ja muuttamaan näitä. Tällä tavoin pystyin siirtämään lisäosia valikosta toiseen ja valikon tasolta toiselle.



KUVA 9. Admin Menu Editorin käyttöliittymä

Paikat, joihin lisäosat asentuvat, ovat täysin sen lisäosan tekijän päätettävissä. Yleensä ne pitäisi laittaa Asetukset-valikon alaisuuteen. Toisinaan tekijä kuitenkin rikkoo tätä käytäntöä asettamalla lisäosan paikaksi jonkun toisen valikon tai jopa ylentämällä sen perusvalikkotasoon. Tässä tasossa sijaitsevat kaikki WordPressin kiinteät valikot ja se saattaa aiheuttaa käyttäjässä hämminkiä.

Admin Menu Editorilla pystyin siirtämään orvoiksi joutuneita lisäosia ja piilottamaan kaikki tuottajille tarpeettomat toiminnot Asetukset-valikon alaisuuteen. Myös tämän valikon piilotin näkyvistä.

5.3.2 April's Facebook Like Button

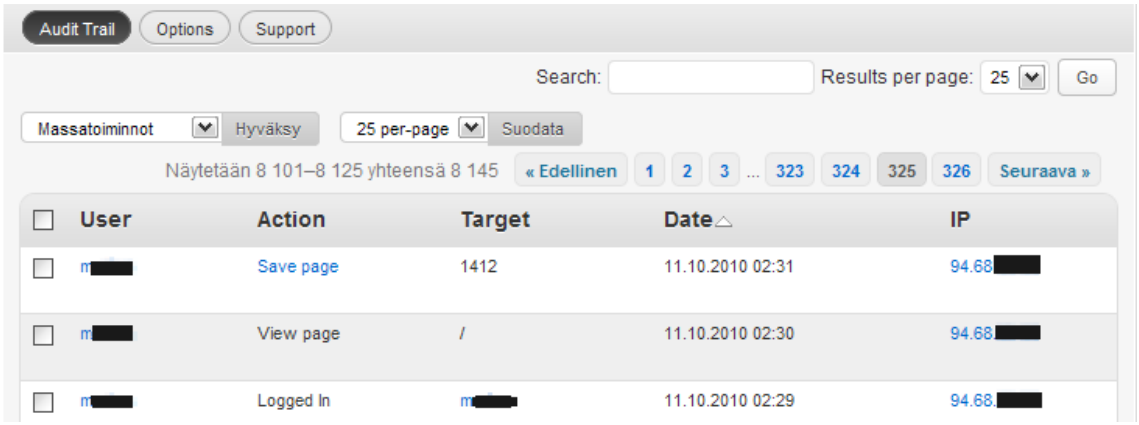
Sivujen pääasiallisen sisällön, eli kuvien ja videon, vuoksi nähtiin hyvänä yhdistää Facebookin tykkäämis-/suosittelu-ominaisuus osaksi artikkelien kommentointi-järjestelmää. Tähän olisin voinut ottaa suoraan koodipätkän Facebookin kehittäjäsivuilta. Vanhan kokemuksen perusteella kuitenkin päädyin asentamaan April's Facebook Like Buttonin (AFLB). Sen käyttöönotto ja käyttöliittymän mahdollistamat helpot ominaisuuksien käyttöönotot toi enemmän ajallista hyötyä kuin itse näiden koodaaminen. Tässä otin myös tietoisen riskin, sillä lisäosaa ei ole päivitetty vuoteen ja sinä aikana Facebook on muuttanut yhteyskäytäntöjään useasti. Toisaalta luotin lisäosan toimintaan juuri tämän vuoksi. Sen toiminta ei ole pettänyt lukuisista muutoksista huolimatta.

Lisäosan asetuksissa oli mahdollista asettaa painike ennen taikka jälkeen artikkelin. Se oli myös mahdollista lisätä manuaalisesti lyhyellä koodin pätkällä. Käyttöliittymän kautta pystyi myös määrittämään muun muassa, näkyykö painike sivuilla, vai pelkästään artikkeleissa, minkä kokoinen se on, mitä verbiä siinä käytetään, sekä minkä kokoinen painike tulee olla. (WordPress, 2010h).

Lisäsin sivuille painikkeen käyttämällä manuaalista tapaa. Tällä tavoin pystyin parhaiten kontrolloimaan painikkeen esiintymistä sekä sen tyyliä.

5.3.3 Audit Trail

WordPress on alustana siitä ikävä, että se on suosittu niin käyttäjien kuin hakkeroijien keskuudessa. Road Dreamin sivut hakkeroitiin ja tätä varten asensin sivuille Audit Trail-työkalun (Kuva 10), jonka avulla kykenin seuraamaan sivuston tapahtumia. Työkalu raportoi reaaliaikaisesti kävijöiden IP-osoitteet sekä erittelee, ovatko he järjestelmän sisällä, vai selaavatko vain sivuja ja kuuluvatko nämä käyttäjät niihin valtuutettuihin, joilla on käyttäjätunnus, vai ovatko he tavallisia vierailijoita. Raporteista myös selviää, mitä he ovat tehneet, jos ovat järjestelmässä sisällä jotain tehneet (WordPress, 2010i).

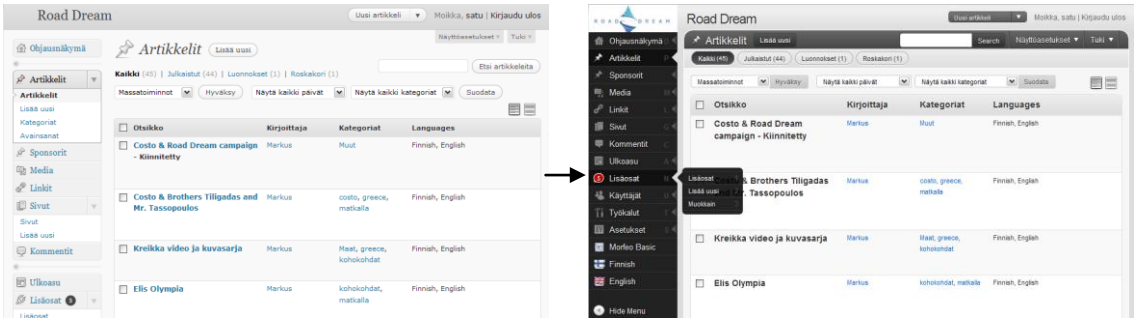


KUVA 10. Audit Trail lisäosan toiminta käytännössä.

Tämän avulla päättelin, mikä IP-osoite kuului tälle haittatekijälle ja pystyin estämään tämän pääsyn sivuille.

5.3.4 Fluency Admin

WordPressin käyttöliittymä on tilaa vievä ja käyttäjäystävällisyyden vuoksi sitä oli muokattava. Päädyin Fluency Admin -lisäosaan ensimmäisellä kokeilulla. Se loi käyttöliittymästä elegantin ja sopi myös sattumalta organisaation väreihin. Lisäsin sivuille myös asiakkaan logon, jotta sivut tuntuisivat omilta niin ulkoa- kuin sisältäpäin. Asiakkaan sisällöntuottajalla oli aikaisempaa kokemusta WordPress-alustasta ja häneltä tuli kiitosta paremmasta, kompaktimmasta käyttöliittymästä, joka oli selkeyttänyt työskentelyä selvästi (Kuva 11).

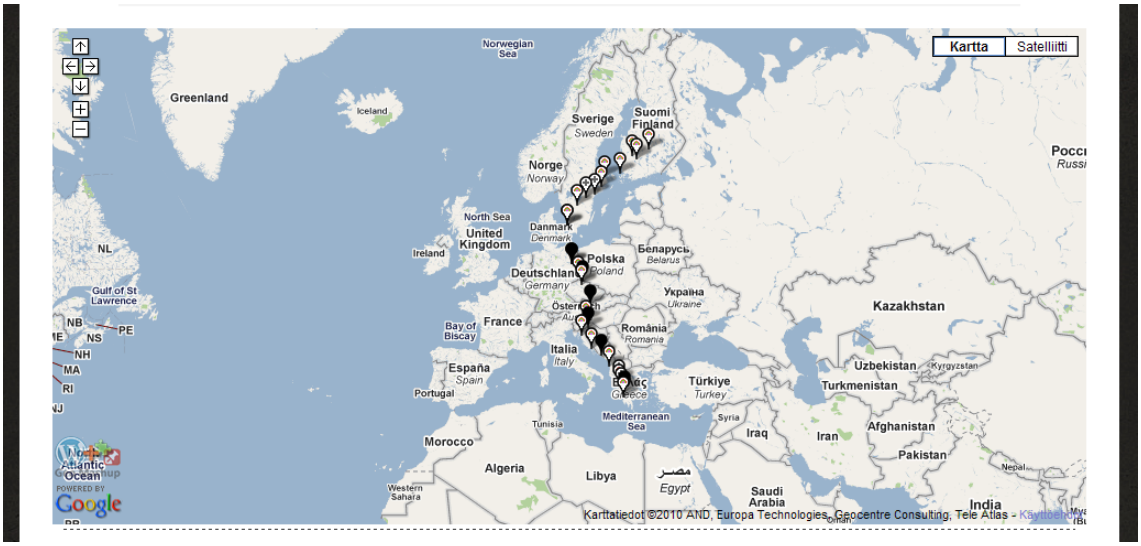


KUVA 11. WordPressin ohjausnäkymän oletusulkoasu ja Fluencyn muokkaama ilme

5.3.5 Geo Mashup

Asiakkaan toiveena oli luoda tai löytää lisäosa interaktiiviselle kartalle, josta kävijä näkisi Road Dreamin matkanetenemisen. Tätä varten löysin oivan lisäosan Geo Mashupin. Tällä työkalulla blogin artikkeleille pystyy määrittämään koordinaatin tarkat sijainnit. Näiden avulla lisäosa osaa asettaa artikkelia kuvaavan paikkamerkin oikeaan kohtaan kartalla. Kun artikkeleita oli kirjoitettu tarpeeksi Road Dreamin blogiin ja ne oli merkitty karttaan, alkoi heidän pyöräilemänsä reitti erottua ja näin kävijä pääsi ikään kuin reaaliaikaisesti seuraamaan heidän edistymistään. Tämä omalta osaltaan osallisti kävijöitä (Kuva 12).

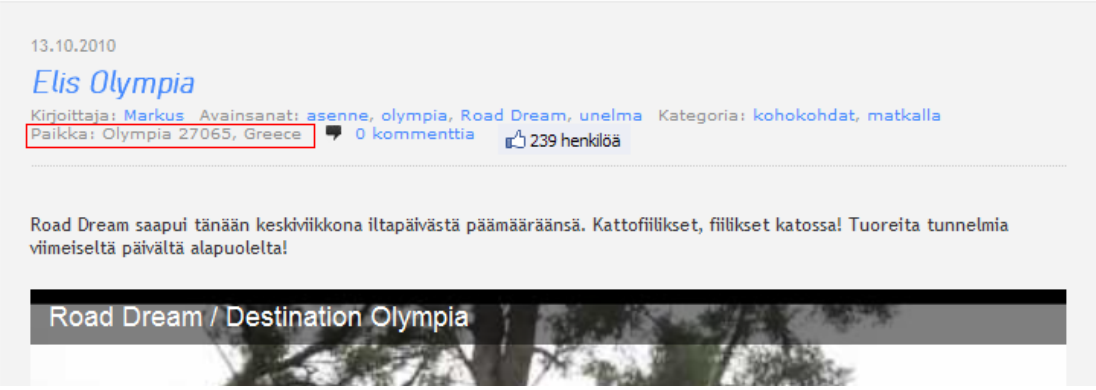
Geo Mashup on työkalu, jolla sisällönhallintajärjestelmään pystytään lisäämään maantieteellinen aspekti. Se käyttää hyväkseen Google Mapsia sekä GeoNamesiä. Google Maps palvelusta se käyttää karttoja ja GeoNamesin avulla nimeää paikat ja etsii mm. nähtävyydet. Tämän lisäosan pystyy lisäämään niin jokaiseen artikkeliin kuin myös erilliselle sivulle. Road Dreamin sivulla Geo Mashupin luoma artikkelireitti on esitettynä erillisenä sivuna (WordPress, 2010j).



KUVA 12. Geo Mashupilla luotu reitti artikkeleiden paikkatietoja hyväksikäyttäen

Tämän työkalun kauneus on siinä, että sen aikaansaamaa dataa on mahdollista käyttää muuallakin. Esimerkiksi Road Dreamin sivuilla käytin paikkatietoja artikkeleiden metatiedoissa. Koordinaatit, aluekoodit ja postinumerot olisi muun

muassa voinut myös esittää, kuten olisi voinut myös listata kaikki kartalla näkyvät artikkelit (Kuva 13). Käyttötapoja on monia.



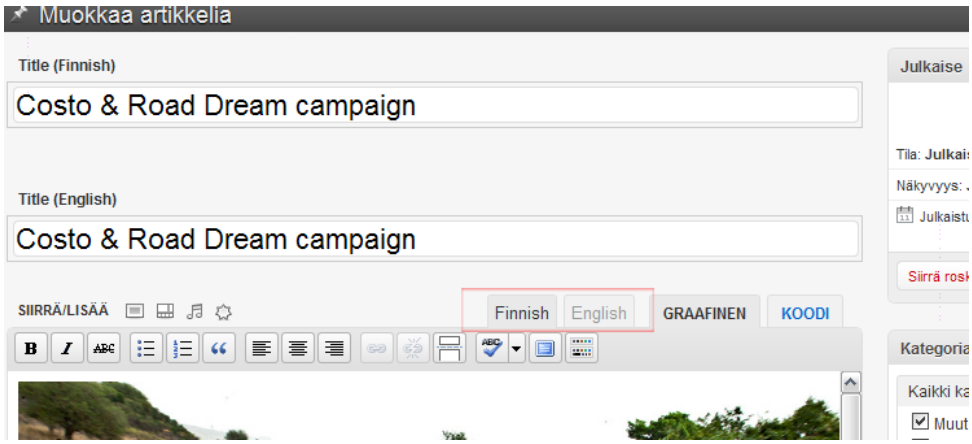
KUVA 13. Geo Mashupilla luotujen paikkatietoja käyttö artikkeleissa

5.3.6 qTranslate

Koska Road Dreamin pääasiallisena tavoitteena oli tehdä elämänsä matka ja dokumentoida se, ei pitänyt unohtaa niitä paikallisia, joita he matkalla tapaisivat. Heti projektin alusta oli selvää, että jonkinlainen järjestelmä kahdelle kielelle oli luotava.

Tutkittuani jälleen kerran lisäosien kirjoa päädyin kokeilemaan qTranslatea. Tämä työkalu osoittautui hyväksi sen monipuolisuuden ja jälleen helppokäyttöisyyden vuoksi. Asiakkaan ei tarvinnut kuin monistaa oma sisältö englannin kielelle ja järjestelmä teki kaiken muun tämän puolesta.

qTranslate loi artikkeleiden luomiseditoriin vaihtoehdot niille eri kielille, jotka oli asetuksista valittu. Asiakkaan tarvitsi vain valita tarvitsemansa kielen välilehti ja kirjoittaa otsikot ja sisällöt valitsemallaan kielellä. Jos jollekin kielelle ei artikkelia käännetä, ei se myöskään silloin näy käyttäjälle, joka sen kielen on valinnut (Kuva 14).

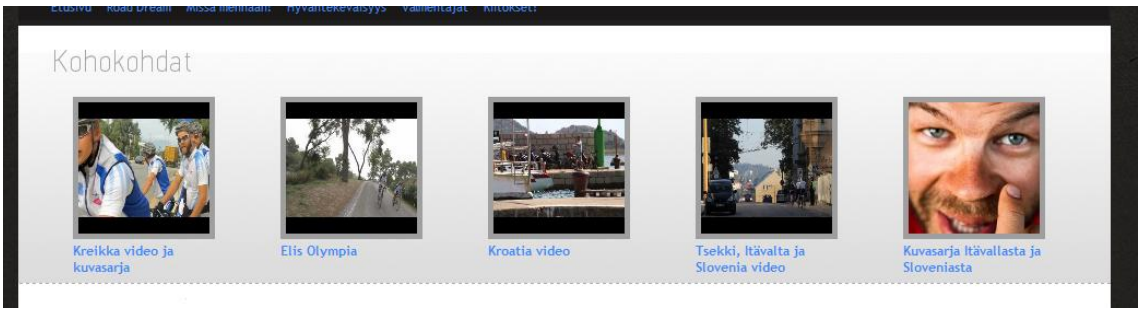


KUVA 14. Monikielituki käytännössä artikkeleiden muokkaussivulla.

Artikkelit eivät ole ainoaa sisältöä, mihin eri kielivälilehdet pystyi luomaan. Myös sivuja pystytään luomaan eri kielille. Ongelmaksi osoittautui itse koodiin lisätty sisältö. Tällaisia Road Dreamin sivuilla oli esimerkiksi artikkelien arkisto. Ongelman ratkaisin luomalla if-elseif-lausekkeen avulla WordPressin omaa syntaksia käyttäen joka tarkastaa käyttäjän valitseman kielen ja näyttää sisällön sen mukaan (WordPress, 2010k).

5.3.7 Youtube Thumbnailer

Omasta aloitteestani ehdotin sivuille Kohokohdat-osiota (Kuva 15), josta näkisi Road Dreamin viisi tuoreinta parasta artikkelia. Tätä varten kokeilin erilaisia lisäosia, joista mikään ei soveltunut omaan tarkoitukseeni, joten koodasin sen itse. Ongelmaksi osoittautui artikkeleihin lisättyjen videoiden pikkukuvien näyttäminen Kohokohdissa. Koska Youtubea käytettiin videotallenteiden lisäämiseksi sivuille, osoittautui Youtube Thumbnailer hyväksi välineeksi. Se luo automaattisesti pikkukuvan Youtube-videoista ja tallentaa nämä palvelimelle. Sitten niitä pystyi käyttämään artikkeleiden kuvina, jolloin ne myös näkyivät omilla paikoillaan Kohokohdat-osiossa (WordPress, 2010l).



KUVA 15. Youtube Thumbnailer lisäosalla saatuja pikkukuvia Kohokohdat-osioissa

5.4 Ongelmat

Road Dreamin sivuja toteutettaessa ei ilmennyt järin suuria ongelmia. Pulmia ja ongelmanratkaisua vaativia tehtäviä kyllä oli, mutta mitään koko järjestelmään vaikuttavia ei ollut. Prosessi oli hyvin helppo ja pahimmat ongelmat syntyivätkin sivujen jo valmistuttua.

5.4.1 Palvelinvaikeudet

Ensimmäiset ongelmat syntyivät hieman ennen uusien sivujen avaamista siirrettyäni sivuston Road Dreamin hankkimalle kotisivutilalle ja testatessani sitä sen natiivissa ympäristössä. Ensinnäkin palvelin oli konfiguroitu niin, että käyttäjän käydessä sivuilla, se ei ymmärtänyt index.php-tiedoston olevan etusivuna vaan se jätti sen täysin huomioimatta haluten käyttää index.html tiedostoa sivuston etusivuna.

Tämä ongelma oli helposti ratkaistavissa. Tätä ongelmaa ei olisi tosin pitänyt koskaan ollakaan ja sitä olisikin voinut pitää eräänlaisena enteenä tulevasta. Ennen virallista julkaisua sekä sen jälkeen huomasin palvelimen toiminnan olevan tahmaista. Toisinaan jopa monen tunnin ajan ja kerran sivut olivat tavoittamattomissa koko päivän ajan. Road Dreamin projektin kannalta epäluotettava palvelin ei olisi tullut kuuloonkaan. He tarvitsivat palveluntarjoajan, joka kykenisi tarjoamaan palvelintilaa palvelimelta, joka olisi todennäköisesti ylhäällä silloin, kun heillä olisi Internet-yhteys käytössään. Eikä niin, että he hakevat ulkomailta Internet-yhteyden, kun sivut näyttäisivät toimivan. Aloin tässä vaiheessa epäilemään palveluntarjoajan kykyä tarjota toimivaa palvelua.

Konsultoin tästä asiasta Road Dreamia ja vähän ennen heidän matkan alkuaan alettiin harkita toista palveluntarjoajaa. Iltalehden uutisen aiheuttaman kävijäryntäyksen ja palvelimen kaatumisen johdosta vaihto toiseen palveluun vahvistui. Uutinen tuli täysin yllätyksenä tiimille, eikä siitä ehditty palveluntarjoajaa varoittamaan ajoissa. Tämän vuoksi heidän mielestään käyttöehtoja oli myös rikottu, sillä niissä palveluntarjoaja vaati saada etukäteen tietää, jos yllättäviä kävijämäärien kasvuja olisi tulossa. Tämä ehto on hyvin vaikea toteuttaa etenkin ulkomailta käsin.

Road Dream hoiti palveluntarjoajan vaihdon ja uusi kotisivutila olikin samana päivänä auki ja domain saatiin ohjattua seuraavana päivänä tähän uuteen tilaan. Tämä palvelin toimi odotetusti. Se ei takkuillut ja vain kerran se oli huoltotöiden vuoksi tavoittamattomissa. Road Dreamin saavutettua Kreikan julkaisi Iltalehti uuden artikkelin, joka aiheutti jälleen kerran uuden kävijäryntäyksen. Palveluntarjoaja reagoikin tähän ärhäkkäästi yrittäen parhaimpansa mukaan pitää sivut kaikkien kävijöiden saatavilla. ”Perjantaina sivuille kohdistui aikamoinen liikenne ja kuormaa jaettiin Helsingin pään laitteemme läpi, jotta pystyimme palvelemaan kaikki sivuille kohdistuvat http-pyynnöt” ilmoitti palveluntarjoaja (Linna 2010). Palvelimella otettiin sivustolle käyttöön reverse proxy, jonka avulla kävijämääristä palvelimelle aiheutunutta kuormaa kyettiin keventämään.

Näistä toimista johtuen WordPressiin kirjautuminen lakkasi toimimasta sekä sivuille tehdyt muutokset ilmestyivät huomattavalla viiveellä. Tätä ongelmaa yritettiin ratkaista kirjautumista ja muokkaamista varten luodulla admin.roaddream.org-alidomainilla. Tuotakaan osoitetta käyttämällä ei kirjautuminen onnistunut, mutta tehdyt muutokset oli mahdollista nähdä reaaliaikaisesti. Nämä muutokset päivittyisivät itse sivustolle omaan tahtiinsa.

Road Dream projekti oli tässä vaiheessa loppusuoralla, eikä sivuille tarvinnut lisätä kuin kaksi artikkelia sekä muokata muutamaa sivua. Tähän keksin ratkaisuksi luoda Road Dreamin sivustosta kopion oman domainini alle. Näin asiakas pystyi kirjautumaan tähän ns. peilisivustoon ja tekemään muutokset siellä. Tämän jälkeen kykenin päivittämään Road Dreamin omat tietokannat näillä uusilla tiedoilla. Nämä muutokset olisin kyllä pystynyt tekemään suoraan tietokantoihin, mutta näin tämän asiakkaan kannalta ystävällisemmäksi ratkaisuksi.

Palveluntarjoajan kanssa päädyttiin siihen, että kirjautumisen kuolemisesta huolimatta pidetään palvelimen asetukset nykyisellään. Sillä sivuston päivitys kyllä onnistui, vaikka kiertoteitä pitkin, mutta yllättävien liikennemäärien vuoksi olisi hyvä, että sivut pystyisivät pystyssä.

5.4.2 Tietoturvaongelmat

WordPress ei ole aukoton järjestelmä. Toisin kuin useita muita www-sisällönhallintajärjestelmiä, WordPressiä päivitetään usein ja tietoturva-aukkoja etsitään aktiivisesti. Tätä 3.0 versiota varten monia aukkoja paikattiin ja tällä hetkellä tilanne on se, että suurin vastuu tämän alustan tietoturvasta on käyttäjällä.

Huomasin yhtenä päivänä, että juuri julkaistun artikkelin loppuun oli ilmestynyt hallinnan puolelta laatikoita ja kenttiä, joita pystyi käyttämään. Poistin nämä kentät hetimiten artikkelista. Illemmalla huomasin sivuilla virheen. Etusivu lataantui aina artikkelin metatietoihin asti, jossa se kohtasi ongelman ja keskeytti lataantumisensa.

Tutkittuani tuon kohdan koodia huomasin, että sieltä oli kadonnut php-tägit tai ne olivat vaillinaisia aiheuttaen erinäisiä ongelmia. Oli huolestuttavaa, että heti etusivulle ilmestyneiden lomakekenttien jälkeen ilmaantui tällaisia ongelmia. Etenkin kun en ollut koskenut tiedostoihin, jotka sisälsivät koodia, johon oli koskettu hetkeen.

6 JATKOKEHITYS

Teeman funktiot sisältävää tiedostoa, joka tässä luotiin, ei voi ihan noin vain lisätä teemaan kuin teemaan. Tällä hetkellä suurin osa ominaisuuksista vaatii kehittäjän väliintuloa. Esimerkiksi kustomoituja sisältötyyppejä ei voida lisätä tuosta vain. Teeman funktioita on siten edelleen muokattava, jotta se sopii uuden projektin tai sivun tarkoituksiin. Tällä hetkellä se on niin sanotusti kalibroitu Road Dreamille. Tätä varten tarvitsisikin luoda jonkinlainen lisäosa käyttöliittymin. Asiakas tai teeman ostaja kun ei useimmiten omaa tarvittavaa tietotaitoa. Tähänhän voisi käyttää kokonaan ulkopuolista lisäosaa, mutta se että tekee itse, antaa vapaammat kädet. Ja on myös helpompi vaikka muokata sitä käyttöliittymää asiakkaan näköiseksi tai antaa jopa asiakkaalle itse käyttöliittymässä mahdollisuus vaikuttaa näihin asioihin. Itse tehty lisäosa on aina parempi myös sen suhteen, että kehittäjä tuntee sen läpikotaisin ja pystyy ongelmatilanteissa helpommin selvittämään ja löytämään ratkaisun ongelmiin. Ulkopuolisten lisäosissa ei myöskään aina ole mahdollista luoda käännöstiedostoja koskematta suoraan sen koodiin. Asiakkaan kannalta on nimittäin parempi asia, että lisäosat ja ominaisuudet ovat tämän omalla kielellä jotta koko sisällönhallintajärjestelmän käyttöliittymä olisi johdonmukainen.

Lisäominaisuuksina teeman funktioihin voisi lisätä käyttäjälle mahdollisuuden muuttaa teemansa taustaväriä tai -kuvaa, kirjasimia, tekstin värejä tai otsakkeen kuvaa. Tämän avulla on myös mahdollista luoda eräänlainen illuusio asiakkaalle tämän mahdollisuuksista vaikuttaa sivujensa ulkoasuun.

7 YHTEENVETO

Asiakas, joka oli jo tutustunut WordPressiin, oli mielissään luomistani hallinnallisista muutoksista. He pitivät Fluency-lisäosan tuomasta uudesta ulkoasusta, sponsorien hallintaan luodusta kustomoidusta sisältötyypistä sekä monikielituesta. Etenkin se, että löysin ja pystyin toteuttamaan nopealla aikataululla heillä toimivan ratkaisun monikielisyydelle oli erittäin tärkeää. He eivät olisi ehtineet eivätkä välttämättä osanneet itse muokata järjestelmästä sellaista kuin mitä minä siitä tein. Tämä ominaisuus kun kuitenkin lisättiin jo matkan alettua.

Projektin työstämisen aikana ilmenneet ongelmat opettivat paljon uusia asioita WordPressistä ja palvelimista. WordPress ei välttämättä ole paras järjestelmä raskasta tietoa, kuten videota, sisältäville sivuille. Se ei ole niin kevyt ja ketterä, etenkään silloin, kun kävijöitä on reilusti enemmän.

Tavoitteet tälle koko projektille olivat hyvin korkealla, sillä projekti ja täten myös itse sivut olivat näkyvissä useaan otteeseen mediassa. Niihin kuitenkin yllettiin. Tiimi pystyi käyttämään järjestelmää tehokkaasti paikasta riippumatta. Heidän vaatima käyttäjäystävällisyys saavutettin. He pystyivät markkinoimaan matkaansa paikallisille sivujen kielituen avulla. Tämä tavoite saavutettiin myös. Matkan reaaliaikainen seuranta kartan avulla onnistui löytämäni ja hyväksikäyttämäni Geo Mashup lisäosan avulla. Teknisesti projekti onnistui harvinaisen hyvin ja kaikki ne ominaisuudet onnistuttiin toteuttamaan, jotka sivuille haluttiin.

Tavoitteet raportin osalta eivät olleet korkealla lähtiessäni tätä kirjoittamaan. Tätä työstäessäni ajatukseni kuitenkin muuttuivat ja panostinkin tähän suunniteltua enemmän. Oli ilo huomata kuinka vaivattomasti kirjoittaminen sujui ja miten paljon aiheestani tiesinkään. T ämä lisäsikin motivaatiotani koko kirjoitusurakkaa kohtaan. Olen tyytyväinen työhöni.

LÄHTEET

Boiko Bob. 2005. Content Management Bible. 2nd Edition. Indiana, USA: Wiley Publishing, Inc.

CMS critic 2010. Interview Dotnetnuke Corp CEO Navin Nagiah. Luettu 18.10.2010 http://www.cmscritic.com/interview-dotnetnuke-corp-ceo-navin-nagiah/

Codex 2010a. The Loop. Luettu 16.10.2010. http://codex.wordpress.org/The_Loop

Codex 2010b. Plugin API: Actions. Luettu 16.10.2010. http://codex.wordpress.org/Plugin_API#Actions

Codex 2010c. Administrator Panels. Luettu 16.10.2010. http://codex.wordpress.org/Administration_Panels

Codex 2010d. Theme Development. Luettu 15.10.2010. http://codex.wordpress.org/Theme_Development

Codex 2010e. Posts Categories. Luettu 15.10.2010 http://codex.wordpress.org/Posts_Categories_SubPanel

Codex 2010f. User Levels. Luettu 15.10.2010 http://codex.wordpress.org/User_Levels

Codex 2010g. Admin Menu Editor. Luettu 20.10.2010 http://wordpress.org/extend/plugins/admin-menu-editor/

Codex 2010h. April's Facebook Like. Luettu 20.10.2010 http://wordpress.org/extend/plugins/aprils-facebook-like-button/

Codex 2010i. Audit Trail. Luettu 20.10.2010 http://wordpress.org/extend/plugins/audit-trail/

Codex 2010j. Geo Mashup. Luettu 20.10.2010 http://wordpress.org/extend/plugins/geo-mashup

Codex 2010k. QTranslate. Luettu 20.10.2010 http://wordpress.org/extend/plugins/gtranslate

Codex 2010l. Youtube Thumbnailer. Luettu 20.10.2010 http://wordpress.org/extend/plugins/youtube-thumbnailer

Codex 2010k. QTranslate. Luettu 20.10.2010 http://wordpress.org/extend/plugins/gtranslate

DocForge 2010. Content Management System. Luettu 16.10.2010
http://docforge.com/wiki/Content_management_system

DPCI 2010. Web Content Management Systems. Luettu 18.10.2010
http://www.databasepublish.com/solutions/web-content-management-systems

Drupal 2010. History. Luettu 17.10.2010
http://drupal.org/about/history

Jasra, Manoj 2007. CMS Watch Releases Enterprise CMS. Luettu 18.10.2010
http://www.webanalyticsworld.net/2007/04/cms-watch-releases-enterprise-cms.html

Joomla 2010. About Joomla. Luettu 18.10.2010
http://www.joomla.org/about-joomla.html

Kaiser, Paul 2010. Introducing WordPress 3 Custom Taxonomies. Luettu 11.09.2010.
http://net.tutsplus.com/tutorials/wordpress/introducing-wordpress-3-custom-taxonomies/

Kampffmeyer, Ulrich 2006. Enterprise Content Management. Köln, Saksa: DFS Druck Brecher GmbH.

Linna, Mika 2010. Sähköposti palvelimen kuormituksesta. Luettu 20.10.2010

Metcalf, Nick 2010. Tutorial: WordPress 3.0 Custom Post Types. Luettu 11.09.2010
http://www.designjuices.co.uk/2010/06/tutorial-wordpress-custom-post-types/

Mixergy 2009. The Biography of WordPress with Matt Mullenweg. Luettu 26.10.2010
http://mixergy.com/the-biography-of-wordpress-with-matt-mullenweg/

MySQL 2010. What is MySQL. Luettu 16.10.2010
http://dev.mysql.com/doc/refman/5.0/en/what-is-mysql.html

Negoita, Cosmin 2010. The Essential Guide to WordPress 3.0 Custom Taxonomies. Luettu 11.09.2010
http://www.1stwebdesigner.com/wordpress/essential-guide-wordpress-custom-taxonomies/

PHP 2010a. What is PHP? Luettu 16.10.2010
http://fi.php.net/manual/en/intro-whatis.php

PHP 2010b. What Can it do? Luettu 16.10.2010
http://fi.php.net/manual/en/intro-whatcando.php

W3C 2010a. HTML. Luettu 16.10.2010
http://www.w3.org/html/

W3C 2010b. CSS. Luettu 16.10.2010
http://www.w3.org/Style/CSS/Overview.en.html

WordPress 2010a. Requirements. Luettu 16.10.2010
http://wordpress.org/about/requirements/

WordPress 2010b. About. Luettu 16.10.2010
http://wordpress.org/about/

WordPress 2010c. Thelonius. Luettu 18.10.2010
http://codex.wordpress.org/Version_3.0

WordPress 2010d. History. Luettu 18.10.2010
http://codex.wordpress.org/History

W-Shadows 2008. Admin Menu Editor for WordPress. Luettu 22.10.2010
http://w-shadow.com/blog/2008/12/20/admin-menu-editor-for-wordpress/